Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1542

Seguimiento de múltiples objetos en un sistema multi-cámara

Autor: Abel Naya Forcano

Directores:
Eduardo Montijano
Ana Cristina Murillo

Trabajo Fin de Máster
Máster en Ingeniería Informática
Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

29/Junio/2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Abel Naya Forcano,

con nº de DNI 73020745-T en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Máster, (Título del Trabajo) Seguimiento de múltiples objetos en un sistema multi-cámara,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 29 de Enero de 2018

Fdo: Abel Naya Forcano

Seguimiento de múltiples objetos en un sistema multi-cámara

Resumen

El seguimiento de objetos basado en visión es un campo dentro de la visión por computador que tiene gran interés en diversos ámbitos, por ejemplo se utiliza como base para identificar peatones desde los sistemas de visión de los coches autónomos o para proporcionar información de interés en los sistemas de vigilancia o monitorización.

Uno de los retos para disponer de un sistema de seguimiento visual robusto consiste en conseguir el correcto funcionamiento e integración de varias tareas, principalmente la detección de los objetos, la asignación de identidades y el seguimiento de éstos. Existen multitud de trabajos al respecto, centrándose y mejorando algunos de los problemas mencionados, pero aún quedan muchos aspectos por resolver y mejoras que hacer para lograr un reconocimiento eficiente y robusto.

Este trabajo se ha enfocado en una de las vías de investigación existentes en este campo para conseguir sistemas más robustos, concretamente en utilizar sistemas multicámara en los que se tienen varias cámaras disponibles que colaboran entre sí para lograr una identificación más exacta, al tener más información disponible. Gracias al uso de múltiples cámaras es posible seguir con mayor facilidad a una persona aunque en alguna cámara ésta se encuentre total o parcialmente oculta. Los principales objetivos del trabajo han sido:

- Realizar un estudio del estado del arte comparando diferentes alternativas ya existentes para una cámara.
- Proponer posibles mejoras a las alternativas existentes hasta lograr una solución multi-cámara, desarrollando nuevos métodos y estrategias para realizar tanto el reconocimiento como el seguimiento de múltiples objetos en escenas mediante el uso de múltiples cámaras con campos de visión superpuestos.
- Realizar diferentes experimentos mediante el uso de datasets existentes específicamente diseñados para este problema, analizando los resultados obtenidos tanto con métricas usadas en el estado del arte como métricas y gráficas propias que han permitido comprobar el desempeño y robustez en diversas situaciones.

Los experimentos realizados muestran que se ha conseguido diseñar un sistema capaz de mejorar el seguimiento de múltiples personas en las escenas gracias al uso de múltiples cámaras, respecto a utilizar únicamente una de ellas. En particular, obtiene resultados más robustos cuando las personas se cruzan en la escena de forma que en al menos una cámara no se produce cruce, o cuando una se oculta pero reaparece poco después. Es el hecho de utilizar varias cámaras el que permite que mientras en una de ellas las personas sean distinguibles esa información se propague al resto.

Índice general

Capítulo 1

Introducción

El seguimiento de objetos basado en visión es un campo dentro de la visión por computador que tiene gran interés en diversos ámbitos. La Figura 1.1 muestra dos ámbitos de aplicación del seguimiento visual, por ejemplo se utiliza como base para identificar peatones desde los sistemas de visión de los coches autónomos o para proporcionar información de interés en los sistemas de vigilancia o monitorización. Gracias a un buen seguimiento se puede analizar la trayectoria de objetos en movimiento y realizar un mapa de su recorrido para analizarlo.

Figura 1.1: Ejemplo de aplicaciones que necesitan algoritmos de seguimiento visual de personas. Conducción autónoma (izq) o Video-vigilancia (dcha) (fuente imágenes: [1] y `https://i.ytimg.com/vi/tCEs05X1jdI/maxresdefault.jpg`).

Uno de los retos para disponer un sistema de seguimiento visual robusto consiste en conseguir el correcto funcionamiento e integración de varias tareas, entre otras la correcta detección e identificación de objetos en imágenes, paso previo que permite centrarse en lo que se va a seguir, la asignación de identidades, para identificar dos posibles objetos como el mismo y juntar sus trayectorias, y conseguir un correcto seguimiento de los objetos en escenas con oclusiones o cambios de escala.

Existen multitud de trabajos al respecto, centrándose y mejorando algunos de los problemas mencionados, pero aún quedan muchos aspectos por resolver y mejoras que hacer, como lograr un reconocimiento eficiente y robusto. Una de las vías de investigación para conseguir sistemas más robustos se centra en utilizar sistemas multicámara, como en el ejemplo de la Figura 1.2. En estos sistemas se tienen varias cámaras disponibles que colaboran entre sí para lograr una identificación más exacta, al tener más información disponible.

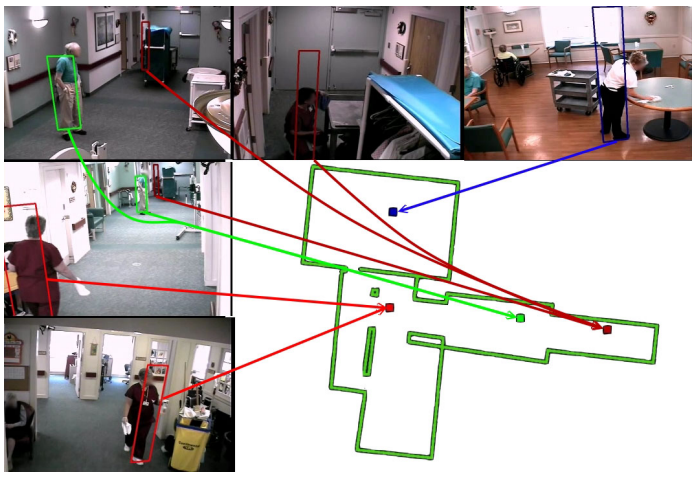

Figura 1.2: Sistema de seguimiento en un edificio mediante un sistema multicámara (fuente: [2])

En este trabajo se ha querido examinar esa vía de investigación con el objetivo de desarrollar e implementar un algoritmo robusto que permita lograr el correcto seguimiento de múltiples personas en una escena mediante el uso de varias cámaras con campos de visión superpuestos (esto es, las cámaras observan la misma escena desde distintos puntos de vista). Para lograr este objetivo también se han estudiado, analizado y probado distintos módulos disponibles diseñados para trabajar con una única cámara (detectores para detectar las personas y seguidores para seguirlas), utilizados tanto como base inicial y comparativa de la solución propuesta como para dar apoyo en las tareas de detección y seguimiento del algoritmo.

1.1. Estado del arte

La tarea del seguimiento de objetos trata problemas de diferentes ámbitos, principalmente la detección de objetos, el seguimiento de éstos y su identificación. En este apartado se proporciona una visión del estado del arte en cada una de estas disciplinas.

1.1.1. Detección de objetos

La tarea de detección de objetos consiste en reconocer tanto la existencia de determinados objetos en una imagen como su posición dentro de ésta. Notar que esta detección es atemporal, se realiza para un instante concreto (una imagen, como la de la Figura 1.3) sin tener en cuenta detecciones realizadas previamente. Este es un problema que ha sido muy estudiado y del que existen muchas técnicas conocidas desde hace tiempo, principalmente con descriptores y algoritmos que tratan de detectar regiones de la imagen en las que con mucha probabilidad se encuentra dicho objeto. Uno de los métodos de este tipo más utilizados es el basado en las características HOG [3].

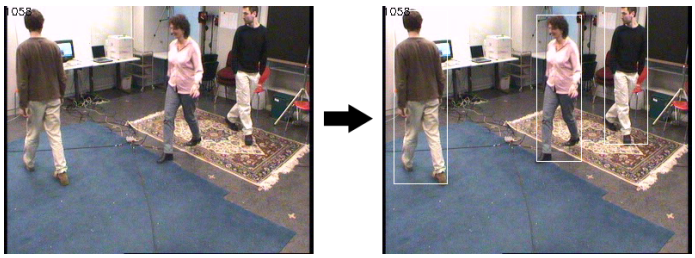

Figura 1.3: Salida esperada de un algoritmo de detección de personas para una imagen dada

Sin embargo, desde hace algunos años, las técnicas basadas en aprendizaje automático o *deep learning* se han popularizado, consiguiendo muy buenos resultados e incluso en la mayoría de casos superando a las técnicas existentes tanto en rendimiento como en tiempo de ejecución. Los detectores basados en redes neuronales utilizan el aprendizaje mediante un conjunto extenso de datos de entrenamiento para aprender y lograr la detección a partir de ejemplos conocidos. Es por esto que normalmente estos detectores están muy condicionados a las características de los datos de entrenamiento utilizados, que suelen ser escenas fijas o con poca variación, logrando excelentes resultados en las tareas concreta para las que han sido entrenados. Un ejemplo concreto sería la detección de peatones en imágenes obtenidas a partir de cámaras colocadas dentro de un coche, en las que el ángulo de la cámara y el campo de vista es muy restringido. En muchos casos si se intentan entrenar para varias tareas genéricas, como por ejemplo detección de personas en cualquier foto tomada con una cámara en cualquier escenario posible, su precisión suele disminuir, aunque existen algunas redes como [4] que funcionan de manera extraordinaria hasta en los casos más extremos, demostrando que este tipo de redes genéricas están mejorando cada vez más y pueden llegar a compararse con otras más específicas.

Aun así, aunque el estado actual de los detectores permite obtener resultados casi perfectos, todavía queda trabajo por hacer y problemas que solucionar. Uno de sus principales inconvenientes es la atemporalidad, pues un detector no tiene en cuenta ejecuciones previas, y por tanto tiene dificultades para detectar objetos que se encuentran total o parcialmente ocultos. Esta es la tarea de los seguidores, que sí aprovechan la información previa.

1.1.2. Técnicas de Seguimiento

Un seguidor (*tracker* en inglés) trata no solo de detectar e identificar objetos en distintos instantes temporalmente ordenados, sino que además debe ser capaz de seguir sus trayectorias tratando de evitar perderlo cuando hay oclusiones. En la Figura 1.4 se muestra un ejemplo de un seguidor. Normalmente se apoyan en el uso de un detector para el reconocimiento de objetos en cada instante de tiempo y en algún algoritmo que permita emparejarlos en trayectorias, por ejemplo el filtro de Kalman [5]. En el caso concreto de seguimiento mediante cámaras se utiliza también con mucha frecuencia similitudes entre imágenes, tratando de buscar los pixeles de la imagen que más se parecen a aquellos entre los que se encontraba el objeto el instante anterior. También es muy usual encontrar algoritmos que tratan de combinar y aprovechar varias técnicas diferentes.

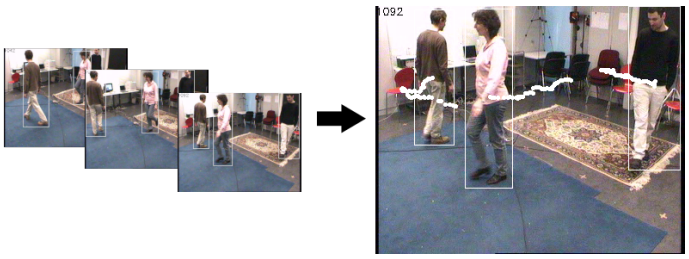

Figura 1.4: Un algoritmo de seguimiento o (*tracker*) es capaz de seguir personas analizando una secuencia de imágenes.

Realizar el seguimiento de un único objeto no suele ser habitual, por ello en el caso de tener múltiples objetos en la escena es necesario identificarlos para poder realizar el seguimiento de cada uno de forma independiente. Esto se puede hacer directamente por cercanía entre detecciones, que tiende a fallar cuando dos o más objetos se cruzan o mediante el uso de características de la imagen, como podría ser color de la ropa [6] o número en la camiseta deportiva [7], entre otros.

Aunque gracias a la detección se logra el seguimiento aun cuando los objetos se encuentran parcial o totalmente ocultos, estos algoritmos se basan en una predicción del movimiento previo mientras el objeto era visible, cuya incertidumbre aumenta con el tiempo si no se tiene información extra para analizar. Si en la escena se conocen los lugares o zonas donde se pierden con frecuencia las detecciones (por ejemplo por la existencia de postes, paredes, etc fijos) se puede intentar aprovechar esta información para mejorar la predicción. Sin embargo cuando se quiere realizar la detección y el seguimiento de múltiples objetos en una cámara, el problema inevitable que aparece es el de las oclusiones ocasionadas por el cruce de los diferentes objetos entre si. En este caso no se puede predecir donde se realizará la oclusión, pues los objetos que ocluyen son los propios objetos que se están siguiendo, dificultando aún más su predicción. Aunque varios algoritmos han intentado resolver el problema mediante modelos predictivos, como en [8], una posible estrategia que no ha sido muy investigada y que se analizará en este trabajo consiste en la utilización de múltiples cámaras con campos de visión superpuestos de forma que todas las cámaras ven la misma escena pero desde puntos de vista diferente, de forma que puedan ayudarse comunicándose entre ellas.

1.1.3. Detección y Seguimiento Multi-cámara

Cuando múltiples objetos interactúan en una escena, tarde o temprano dos de ellos se cruzarán delante de la cámara, provocando la oclusión de uno de ellos lo que puede llevar a perderlo. Incluir varias cámaras que observen la escena desde diferentes puntos de vista permite que aunque en una de ellas los objetos se crucen, probablemente en otra no lo hagan. En esta situación la cámara en la que los objetos no se han cruzado puede ayudar a la cámara en la que sí lo han hecho. Un algoritmo con estas características debe ser capaz de detectar esta situación tratando la información de las diversas cámaras, conociendo la relación existente entre ellas (por ejemplo la posición y rotación de cada una en la escena) identificando las zonas comunes y relacionando los objetos detectados entre ellas.

La propuesta de este trabajo consiste en investigar este tipo de algoritmos y de proponer una solución que aproveche toda la información disponible mediante el uso de varias cámaras para mejorar la detección, con respecto al uso de una.

Encontramos trabajos recientes como [9] que proponen un sistema distribuido en el que las diferentes cámaras se ejecutan en nodos independientes y se comunican con sus más cercanas mediante mensajes. A diferencia de este trabajo nuestro sistema es centralizado, es decir, suponemos que existe un nodo central que recibe la información de todas las cámaras, ver Figura 1.5. De esta manera conseguimos centrarnos en los problema de la detección y la identificación multicámara, permitiendo una mayor velocidad de respuesta y tratamiento completo de los datos.

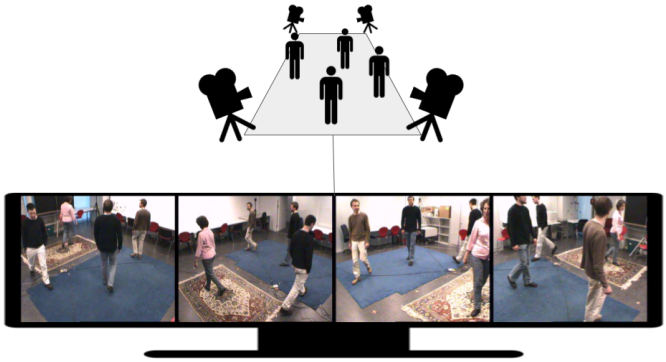

Figura 1.5: Sistema multicámara centralizado, en donde la información de todas las cámaras se analiza en conjunto.

1.2. Objetivos, tareas y alcance

El objetivo principal de este trabajo ha sido desarrollar nuevos métodos y estrategias para realizar el reconocimiento y seguimiento de múltiples objetos en escenas mediante el uso de múltiples cámaras con campos de visión superpuestos. Se ha trabajado con datasets existentes específicamente diseñados para este problema. También se ha realizado un estudio del estado del arte, comparando diferentes alternativas existentes para una cámara y proponiendo posibles mejoras a las mismas hasta lograr una solución multi-cámara.

En cuanto a la metodología utilizada, se ha seguido la típica en proyectos de investigación. Se ha realizado un estudio de la literatura y métodos existentes sobre el reconocimiento y seguimiento de múltiples objetos en escenas mediante el uso de múltiples cámaras, así como de las herramientas utilizadas. Se han evaluado modelos existentes para los datos disponibles, estudiando diferentes métricas, y se han propuesto e implementado mejoras de dichos modelos, evaluando sus prestaciones. Para comprobar el seguimiento se han realizado reuniones semanales con los directores y otros compañeros.

La distribución de tareas aproximada que se ha realizado ha sido la siguiente (ver diagrama de Gant de la Figura 1.6):

- Se ha estudiado y leído la bibliografía relacionada con el problema de reconocimiento y seguimiento de varios objetos con varias cámaras, que ha sido proporcionada por los tutores.
- Se han evaluado empíricamente las diversas soluciones existentes en sistemas de una cámara para la detección y el seguimiento, utilizando implementaciones disponibles en librerías y analizando la mejor de todas ellas.
- Se han propuesto e implementado mejoras y modificaciones en las soluciones de una cámara para lograr un algoritmo que aproveche la información de múltiples cámaras. En concreto se ha diseñado y programado un algoritmo desde cero, salvo por el uso de los mejores módulos encontrados para la detección y el seguimiento.
- Se ha experimentado, comparado y analizado los resultados cuantitativos de la propuesta.
- Se ha documentado el código y el trabajo mediante la redacción de esta memoria.

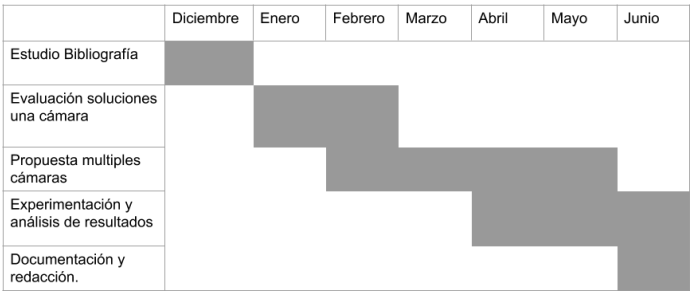

Figura 1.6: Diagrama de Gant de las tareas realizadas

1.3. Entorno de trabajo

Este trabajo se ha llevado a cabo dentro del grupo de robótica, percepción y tiempo real (RoPeRT), un grupo de investigación ubicado en el Instituto de Investigación en Ingeniería de Aragón (I3A).

Las herramientas utilizadas han sido el lenguaje de programación Python, junto con librerías especializadas como OpenCv para el procesado de imágenes y gráficos. También se ha utilizado el entorno caffe2 para ejecutar detectores basados en redes neuronales.

1.4. Resumen de la Memoria

En el capítulo 2 se desarrollará el sistema propuesto, tanto su formulación teórica como su implementación. En el capítulo 3 se explicará más en detalle la arquitectura utilizada. En el capítulo 4 se presentarán los resultados obtenidos. Y en el capítulo 5 se muestran las conclusiones y el trabajo futuro.

Capítulo 2

Seguimiento de personas multicámara

El objetivo principal de este trabajo ha sido la propuesta y posterior desarrollo de un algoritmo centralizado capaz de aprovechar la información de varias cámaras para lograr un seguimiento de múltiples personas en una misma escena. En este capítulo se detalla la formulación del algoritmo, así como los componentes que requiere para su funcionamiento.

2.1. Descripción del problema y solución propuesta

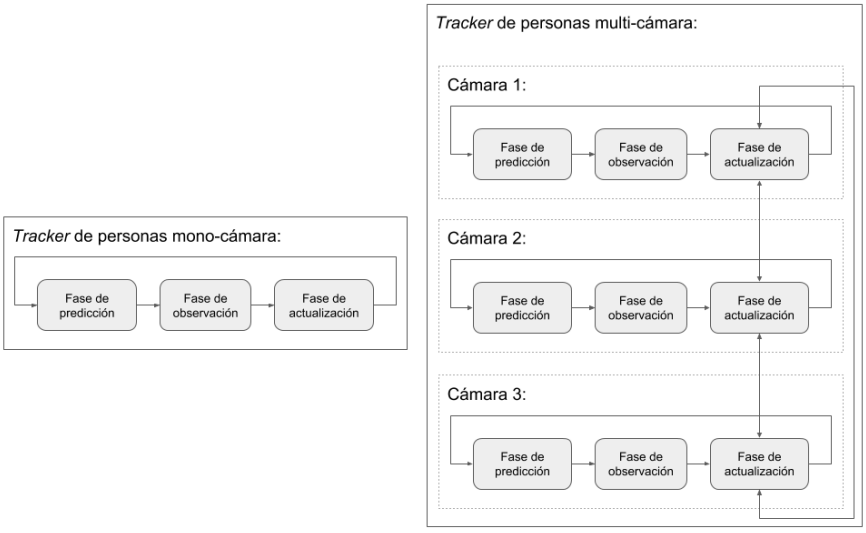

Figura 2.1: Esquemas de un seguimiento mono-cámara convencional (izq) y del sistema multicámara diseñado en este proyecto (dcha), donde los sub-sistemas de cada cámara se intercambian información en la fase de actualización.

A alto nivel el algoritmo funciona idéntico a un seguidor de personas de una cámara con la diferencia de que al utilizar varias cámaras la información de cada una de ellas puede afectar al resultado de las otras, tal y como se observa en la Figura 2.1.

Por esta razón el algoritmo consta de las siguientes tres fases, estructura habitual de los observadores de estado y estimadores:

1. Fase de Predicción: En esta fase se trata de predecir la posición de la persona utilizando únicamente un modelo de comportamiento, estimando la posición a partir de los instantes anteriores. En nuestro caso concreto esta es la fase en la que se ejecuta un algoritmo existente de seguimiento mono-cámara, que 'predice' la posición de la persona.

2. Fase de Observación: Esta fase trata de utilizar la información de algún observador externo capaz de medir la posición real de la persona con cierto margen de error. En nuestro sistema se utiliza un detector de personas existente que devuelve zonas en la imagen donde cree que existen personas.

3. Fase de Actualización: En esta fase se combina la predicción proporcionada por el modelo (fase de predicción) junto con la proporcionada por el detector (fase de observación) con el objetivo de refinar el resultado y proporcionar la posición esperada final. En nuestro sistema, al tener información no solo de la propia cámara sino de otras, es en esta fase donde toda la información obtenida se 'pone en común' para mejorar aun más el resultado. Gracias a este intercambio de datos es posible que una cámara en la que una persona esté oculta pueda ser seguida correctamente si en otras cámaras no lo está.

2.2. Algoritmo de seguimiento multicámara

En esta sección se detallan los elementos comunes del algoritmo diseñado en este trabajo para el seguimiento. El objetivo del algoritmo es el seguimiento de múltiples personas $i \in \mathbb{N}$ mediante el uso de distintas cámaras $j \in J$ en diferentes instantes de tiempo $t \in \mathbb{N}$.

2.2.1. Salida del algoritmo

El algoritmo propuesto sirve para detectar la posición de múltiples personas en una escena a través de varias cámaras. Por esta razón el resultado del algoritmo en cada instante de tiempo t es una lista de rectángulos (regiones o *bounding boxes*) para cada una de las cámaras que indican la posición de cada persona en coordenadas de la imagen: $r_{i,j}(t)$ (en particular las coordenadas de la esquina superior izquierda e inferior derecha) como se muestra en la Figura 2.2.

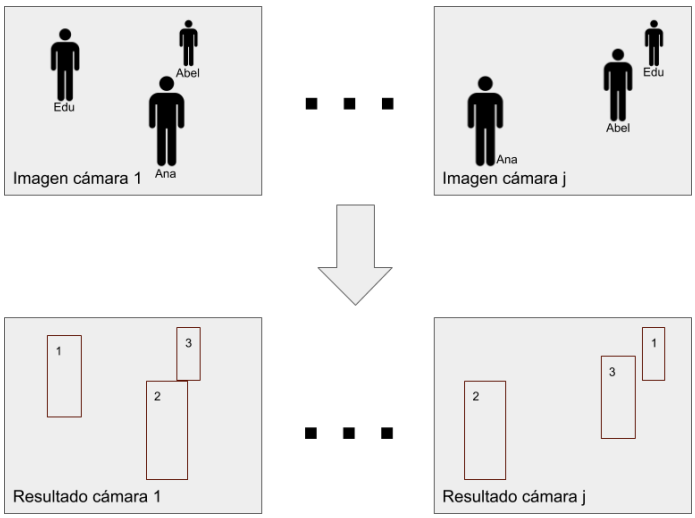

Figura 2.2: Entrada (imagen) y salida (región de la imagen donde se encuentra cada persona) para cada cámara del sistema propuesto de seguimiento, en un instante concreto de tiempo.

2.2.2. Cámaras

El algoritmo está diseñado para trabajar con un número variable J de cámaras. En el caso particular en que $J = 1$, una cámara, el funcionamiento es muy similar a los algoritmos existentes. Cada una de las cámaras debe proporcionar una imagen RGB para un instante concreto, también llamado *frame* $Im_j(t)$, y además deben estar sincronizadas de forma que $Im_1(t), ..., Im_{nc}(t)$ debe corresponder a imágenes del mismo instante de tiempo.

Al trabajar con varias cámaras para un sistema centralizado es necesario tener una manera de juntar la información proporcionada por cada una de ellas en un setup común. En particular, por el tipo de operaciones que se realizan, este setup común corresponde al plano del suelo de la escena, cuyas posiciones de cada persona en cada cámara se representan con coordenadas del plano $p_{i,j}(t)$. Resulta necesario ser capaz de transformar coordenadas en la imagen a coordenadas del suelo. Por esto el algoritmo requiere alguna función que sea capaz de realizar esta conversión, por ejemplo mediante el uso de matrices de homografía. Se establece así una relación no biyectiva entre las regiones en la imagen y las posiciones del suelo: $p_{i,j}(t) = H * f(r_{i,j}(t))$ siendo f la función que extrae el punto inferior medio del rectángulo (se puede ver un ejemplo en la Figura 2.3).

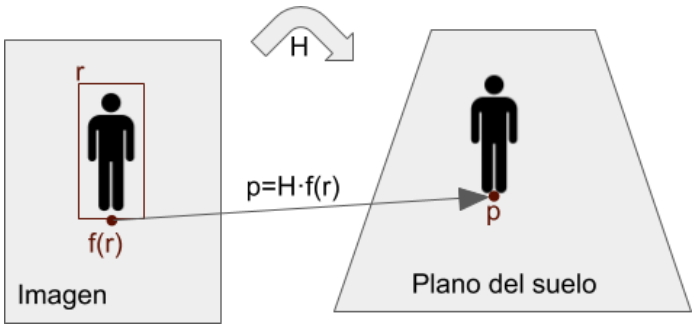

Figura 2.3: Proyección en el plano común del suelo, p, del punto medio inferior de la región de una persona en una imagen, $f(r)$, mediante la homografía H que relaciona el plano de la imagen y el plano del suelo

2.2.3. Módulos necesarios

El algoritmo ha sido diseñado para trabajar en la interacción con múltiples cámaras, para lo que se requiere la existencia de funciones o módulos que trabajan con una cámara. En concreto se necesitan dos módulos:

Un detector mono-cámara que sea capaz de detectar personas en una imagen Im devolviendo una lista de regiones r (de tamaño variable, puede estar vacía) $R = detect(Im)$, ver ejemplo en la Figura 2.4.

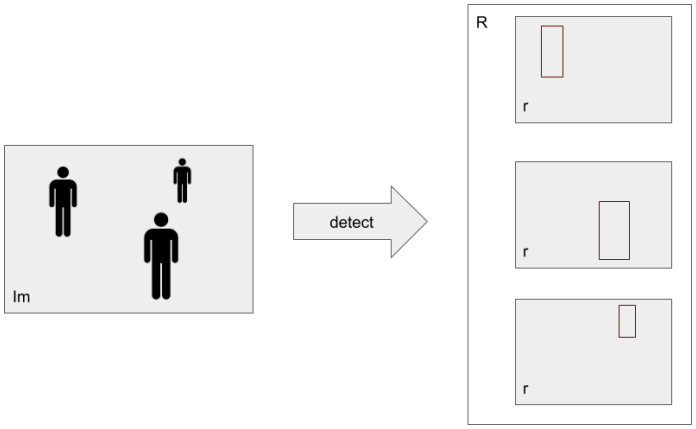

Figura 2.4: Módulo de detección (*detect*): dada una imagen de entrada (Im), la salida (R) es una lista de regiones con probabilidad alta de contener una persona.

Y un seguidor (*tracker*) mono-cámara que sea capaz de devolver una predicción de la posición de una persona en una imagen a partir de la posición inicial en un instante previo y todas las imágenes previas: $r(t) = TR(Im(t), Im(t-1), ..., Im(0), r(0))$. Para simplificar la formulación se asume que un tracker dado TR es inicializado con la región de interés y almacena el estado de las ejecuciones previas, con lo que en esta situación la función se simplifica a $r(t) = TR(Im(t))$. Notar que es posible que el tracker decida que la persona ya no se encuentra en

la imagen, en cuyo caso no devuelve una región válida.

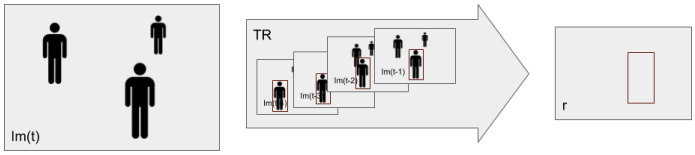

Figura 2.5: Módulo de seguimiento (*tracker*): dada una nueva imagen de entrada (Im), la salida (r) es una región donde cree que se encuentra la persona que se está siguiendo. Almacena el resultado para posteriores ejecuciones.

Ambos módulos no se han implementado directamente, en su lugar se han utilizado implementaciones ya existentes en librerías que proporcionan la funcionalidad requerida.

Por otra parte, para poder comparar la similitud entre regiones, se hace necesario el uso de una medida que indique como de similar son dos regiones dadas $r1, r2$. Siguiendo el uso común en este tipo de algoritmos se ha implementado la función $IOU(r1, r2) \in [0, 1]$ (*Intersection Over Union*) que se calcula dividiendo el área de la intersección de ambas regiones entre el área de la unión. Si ambas regiones son idénticas la intersección es igual a la unión y por tanto $IOU = 1$, en caso de ser disjuntas la intersección es nula y por tanto $IOU = 0$.

2.2.4. Parámetros internos del algoritmo

Para cada persona i en cada cámara j se tiene la posición $p_{i,j}$ y su equivalente región $r_{i,j}$, así como un tracker asociado $TR_{i,j}$. Además de todo esto, el algoritmo requiere la siguiente información extra:

- Conocer durante cuantos *frames* esa persona ha sido detectada correctamente tanto por el tracker como por el detector. En particular se define la variable $\lambda_{i,j} \in \mathbb{Z}$ que indica la duración en *frames* que la persona ha sido detectada con éxito (si $\lambda_{i,j} > 0$) o ha sido perdida ($\lambda_{i,j} < 0$). Se utiliza una misma variable para ambas situaciones ya que éstas son excluyentes. Cuando se indique que una persona se 'marca como encontrada' significa que se aumenta en una unidad esta variable forzando que sea positiva, y del modo opuesto cuando se 'marca como perdida' se resta una unidad, forzando que sea negativa, tal y como se muestra en la ecuación 2.1.

$$\lambda_{i,j} = \begin{cases} \text{máx}(1, \lambda_{i,j} + 1) & \text{Si encontrado} \\ \text{mín}(-1, \lambda_{i,j} - 1) & \text{Si perdido} \end{cases} \tag{2.1}$$

- Conocer si la persona ha sido detectada suficientes veces o no, que llamaremos 'persona validada', pues éstas son más probables de ser personas reales y no detecciones esporádicas de sombras o errores en los módulos. Para esto definimos una variable binaria $v_{i,j} \in \{0, 1\}$ siendo 1 cuando la persona se considera 'válida' y 0 cuando no.

2.3. Formulación del algoritmo

El algoritmo se ha estructurado en 3 fases, predicción, observación y actualización. A continuación se detalla lo que se realiza en cada una de ellas:

Fase de predicción:
La fase de predicción trata de obtener una estimación de la posición de cada persona a partir de la ejecución del módulo tracker, un módulo existente ajeno al algoritmo. En particular se tiene que para cada persona activa i, j se actualiza la nueva región tal y como se define en la ecuación 2.2.

$$r^p_{i,j}(t+1) = \begin{cases} TR_{i,j}(Im_j(t)) & \text{Si encontrado} \\ r^a_{i,j}(t) & \text{Si perdido} \end{cases} \tag{2.2}$$

Además, si el tracker devuelve un resultado válido éste se marca como encontrado. En caso contrario, si el tracker decide que la persona ya no es visible y no devuelve un resultado válido, se marca como perdido.

Fase de observación:
En la fase de observación se toma la lista de observaciones proporcionada por el detector para un instante dado en una cámara: $R_j(t)$ y se actualizan las predicciones previas con la más similar dentro de cierto threshold (τ_{IOU}) tal y como se describe en la equación 2.3.

$$r^o_{i,j}(t+1) = \begin{cases} \arg\max_{r \in R_j(t)} IOU(r, r^p_{i,j}(t+1)) & tq \quad IOU > \tau_{IOU} \\ r^p_{i,j}(t+1) & \text{Si no solución} \end{cases} \tag{2.3}$$

Del mismo modo se marca como encontrado o no según si existe solución a la función anterior o no.

Por otra parte, cuando una de las regiones de $R_j(t)$ se utiliza, ésta se marca (independientemente de cuantas veces se utilice) de forma que se divide el conjunto $R_j(t)$ en dos subconjuntos disjuntos $R_j(t) = U_j(t) \sqcup O_j(t)$ siendo $U_j(t)$ el subconjunto de las regiones que han sido utilizadas y $O_j(t)$ aquellas que han sido omitidas. Esta distinción resultará útil en fases posteriores.

Es importante destacar que esta fase sólo se realiza cuando el detector es ejecutado. Cuando la información no está disponible se mantienen las regiones previas $r^o_{i,j}(t+1) = r^p_{i,j}(t+1)$ y sin cambios en la variable λ.

Fase de actualización:
La fase de actualización engloba varios pasos o subfases, cada uno propuesto para tratar una situación en particular. Se distinguen dos partes: Actualizaciones mono-cámara y multi-cámara.

El primer paso mono-cámara es eliminar los seguimientos de las personas que han sido perdidas durante suficiente tiempo, es decir, eliminar las parejas (i, j) para las que se cumple $\lambda_{i,j} < -\tau_{LOST}$. De esta manera si una persona ha dejado la escena, o tanto el detector como el tracker no son capaces de seguirla, se quita.

El segundo paso mono-cámara consiste en la creación de nuevas personas detectadas que se seguirán. Esto se ha realizado mediante el conjunto $O_j(t)$

consistente en aquellas detecciones proporcionadas por el detector (cuando es ejecutado) que no han sido asignadas a ninguna persona existente. Para cada región $r \in O_j(t)$ se genera un nuevo id i, se inicializa un tracker que se encargue de seguirla, y se asigna a una nueva detección con $\lambda_{i,j} = v_{i,j} = 0$.

Se ha comprobado que en muchas ocasiones, sobre todo cuando hay múltiples personas, los detectores suelen dar falsos positivos. Para mitigar este problema se ha añadido una comprobación adicional multi-cámara: para cada posible nueva detección ésta se proyecta en el plano del suelo y de ahí al resto de cámaras, mediante la homografía inversa (notar que se obtienen puntos en las imágenes, no regiones). Si en alguna cámara este punto no se encuentra cerca de una detección existente se omite (o dicho de otra forma: sólo se admite si las proyecciones caen cerca de detecciones o fuera del campo visual).

Para las partes multi-cámara es necesario definir una variable adicional: el centro de cada persona activa (Figura 2.6). Ésta se define como la media ponderada de las posiciones en el plano de cada posición en cada cámara, como se observa en la ecuación 2.4.

$$\begin{aligned} S_i &= \{j | \exists p^u_{i,j}(t+1) \wedge v_{i,j} = 1\}, \\ p_i(t+1) &= \frac{\sum_{j \in S_i} \lambda_{i,j} p^u_{i,j}(t+1)}{\sum_{j \in S_i} \lambda_{i,j}}. \end{aligned} \tag{2.4}$$

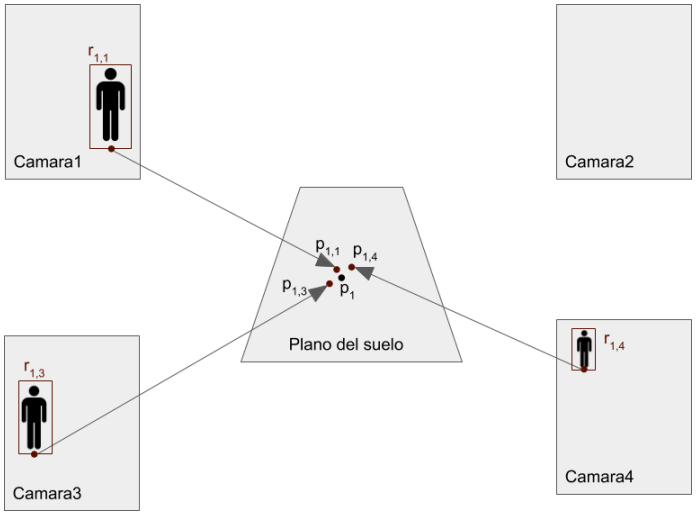

Figura 2.6: Cálculo del centro de una persona en el plano del suelo a partir de las detecciones en las cámaras.

Para el cálculo del centro se utilizan únicamente las posiciones conocidas, que se ponderan por el tiempo que han estado activas, de forma que una cámara que ha reconocido durante más tiempo a la persona tiene mayor peso que otra que la acaba de detectar. Ahora que tenemos el centro de cada persona activa se procede a reasignar ids en función de la cercanía de cada persona a cada grupo. En concreto se tienen dos situaciones:

Si la distancia euclídea en el plano del suelo de una persona activa en una cámara al centro del grupo $d_{i,j} = dist(p_i, p^u_{i,j})$ es mayor que cierto threshold

$d_{i,j} > \tau_{DISTMIN}$, ésta se elimina de ese grupo y se le asigna un nuevo id no utilizado i, marcando además como persona no activa $v_{i,j} = 0$.

Por otra parte, si la menor distancia desde este punto a cualquiera de los centros es menor que cierto threshold, entonces se le asigna ese identificador (ver ecuación 2.5).

$$i^* = \arg\min_m dist(p_m, p_{i,j}) \quad tq \quad dist < \tau_{DISTMAX}, \quad i^* \neq i \Rightarrow i = i^* \tag{2.5}$$

Estas dos situaciones sin embargo no se realizan instantáneamente. En su lugar es necesario que se cumplan las condiciones durante varios *frames* seguidos, tras los cuales se efectúa.

El último paso de todos consiste en marcar una persona como 'válida' tras su correcta detección durante varios *frames* seguidos, definido en la ecuación 2.6.

$$v_{i,j} = 1 \quad \text{Si} \quad \lambda_{i,j} > \tau_{PERSONA} \tag{2.6}$$

Capítulo 3

Arquitectura del Sistema de Seguimiento Desarrollado

En este capítulo se describe la implementación que se ha realizado para poner en marcha el sistema de seguimiento multi-cámara descrito en el capítulo anterior.

3.1. Resumen del sistema

Se ha desarrollado en Python, y las principales bibliotecas externas utlizadas han sido OpenCV (versión 3.3.1 para Python) y Caffe2 (versión 0.8.1). El sistema está organizado en distintos módulos, que corresponden con las distintas funcionalidades y herramientas que hacen falta tanto para la ejecución como para la evaluación del algoritmo propuesto. Cada uno de estos módulos se centra en una de las tareas concretas del sistema, con poco entrelazado entre ellos salvo el necesario, con lo que pueden ser modificados de forma independiente si se quiere utilizar un tracker más avanzado, un detector externo o leer los datos del dataset desde una ubicación en red, por ejemplo. El diagrama de la Figura 3.1 muestra estos módulos y sus relaciones.

Además, se han desarrollado varias herramientas y utilidades externas que, aunque no son necesarias para la ejecución del sistema (y por tanto no se incluyen en la vista de módulos), ayudan en la visualización de los distintos datos. En concreto una herramienta para comprobar la calibración entre las distintas cámaras (Figura 3.3) y dos herramientas para visualizar los datos devueltos por el detector y el groundtruth en cada imagen.

En el anexo B se detalla la puesta en marcha y uso del sistema, así como los pasos necesarios para su instalación.

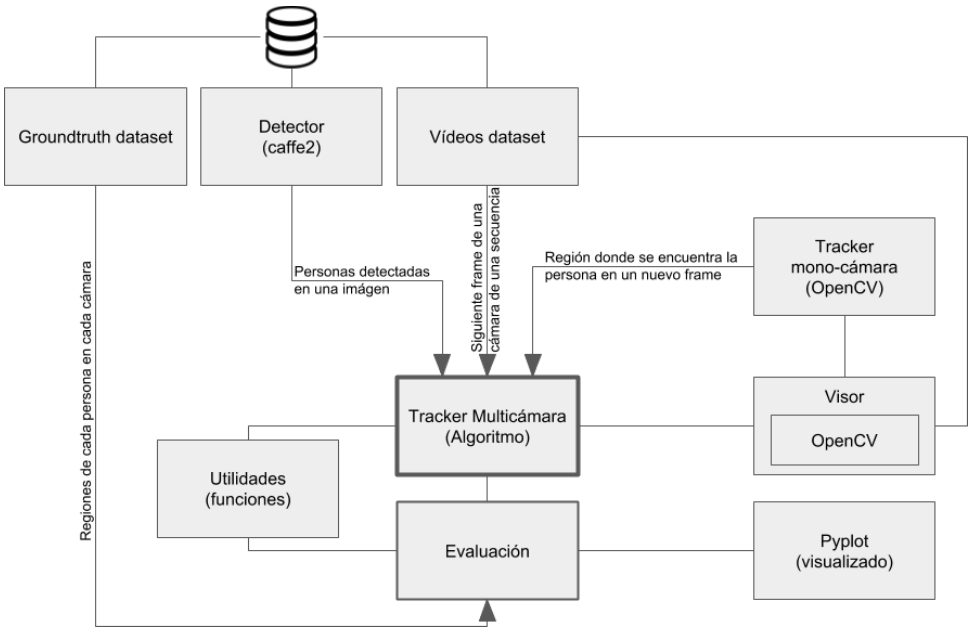

Figura 3.1: Diagrama de módulos del sistema implementado

3.2. Descripción de módulos

A continuación, se describe la funcionalidad de cada uno de los módulos mostrados en el diagrama de la sección anterior:

- ***Tracker* multicámara:** Contiene la implementación del algoritmo desarrollado. Al ser un algoritmo secuencial no ha sido necesario separar su implementación en diferentes clases, aunque por comodidad se han separado algunas funciones en módulos independientes.
- **Evaluación:** Contiene los métodos utilizados para analizar los resultados del sistema y compararlos con el *groundtruth*, tal y como se describe en el capítulo 4.
- **Pyplot:** Biblioteca de Python utilizada para generar y visualizar las gráficas de los resultados (se puede ver un ejemplo en la figura 4.2).
- **Visor:** Este módulo actúa como un *wrapper* del módulo OpenCV, al cual se le ha añadido la funcionalidad de permitir 'retroceder' en el historial de imágenes mostradas, para facilitar el análisis de los resultados (se puede ver una captura de la interfaz en la figura 3.2).
- ***Tracker* mono-cámara:** Los *trackers* utilizados como base, que se encuentran implementados en la biblioteca de OpenCV.
- **Detector:** El detector utilizado corresponde a una implementación realizada en Caffe2. Su ejecución se realizó offline y se almacenaron los resultados en ficheros, que se utilizaron en su lugar mediante un módulo que se encarga de su lectura.

- **Videos dataset:** Módulo que permite devolver los vídeos (en particular las imágenes de cada frame) del dataset correspondiente. Su lectura se realiza a través de OpenCV.
- ***Groundtruth* dataset:** Módulo que permite devolver el *groundtruth* de cada cámara de una escena concreta, información que se lee de fichero.
- **Utilidades:** Conjunto de funciones para cálculos matemáticos o administración de datos, entre otros.

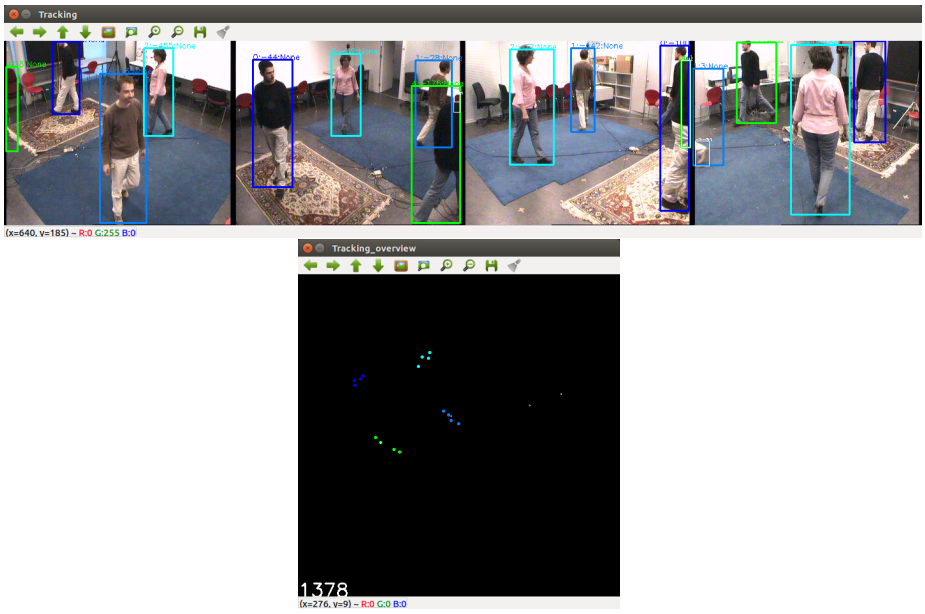

Figura 3.2: Ventanas del sistema en ejecución para el dataset *Laboratory*. Arriba: Vista de las detecciones (regiones) de las personas en las distintas cámaras. Abajo: La posición (puntos) de las personas proyectada en el plano del suelo.

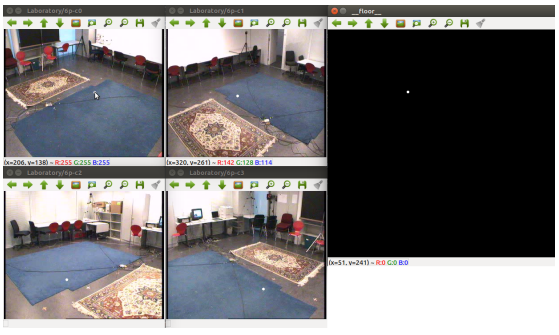

Figura 3.3: Herramienta para comprobar la calibración entre las distintas cámaras. Al mover el ratón en cualquiera de las vistas su posición (punto blanco) se reproyecta en el resto de cámaras y en el suelo.

Capítulo 4

Evaluación del sistema propuesto

En este capítulo se detallan los resultados de la ejecución del sistema, los experimentos realizados y la conclusiones obtenidas. El objetivo del trabajo es el desarrollo de un algoritmo robusto que utilice la información de varias cámaras para mejorar el resultado de detección y seguimiento de personas respecto al uso de una. Los experimentos realizados están orientados a validar la consecución del mismo.

4.1. Configuración de experimentos

Para el desarrollo de los experimentos se ha utilizado un dataset y métricas existentes y comúnmente utilizados en problemas de este área. A continuación se explican los detalles de cada uno.

4.1.1. Dataset

El sistema propuesto en este trabajo está diseñado para trabajar con un sistema multicámara en un entorno con varias personas moviéndose en él, de forma que cada cámara es capaz de visualizar la escena desde un ángulo diferente. Además, es necesaria la existencia de calibración del sistema entre las cámaras y el plano del suelo de la escena, es decir, una función que relacione las coordenadas 2D de puntos en las imágenes de cada cámara con las coordenadas 2D en el suelo.

Por lo tanto, en esta evaluación se ha utilizado un dataset público que cumple estos requisitos, "*EPFL*", creado también para comprobar la mejora de un sistema multicámara y originalmente usado para un detector basado en *background substraction* [10]. Este dataset está disponible en la web de sus autores, incluyendo información de calibración y anotaciones de referencia. Para algunas de las secuencias, están disponibles las matrices de homografía que relacionan cada cámara con el plano del suelo. Para todo el dataset, están disponibles las anotaciones de referencia (o *ground truth*) realizadas a mano, indicando la posición de cada persona en coordenadas del suelo cada 25 frames. Aunque este

ground truth original no es adecuado para nuestro sistema (nuestro sistema devuelve como resultado regiones en las imágenes, y necesitaría que el *ground truth* sea igual), los autores de [11] han generado un *groud truth* adicional a partir del anterior en el que se devuelve la región en cada cámara de cada persona para todos los frames, exactamente lo que se requería para la correcta evaluación de nuestro sistema.

El dataset contiene 5 secuencias diferentes, pero dos de ellas ('Basketball' y 'Passageway') tuvieron que ser descartadas por no tener disponibles o ser erróneas las matrices de homografía. Las 3 secuencias utilizadas que sí tienen disponible toda la información de calibración y anotaciones necesarias, ver Figura 4.1, son:

- 'Terrace': 4 cámaras en las esquinas de una terraza, iluminación exterior. En total se muestran 7 personas entrando y saliendo de la escena durante un total de 3 minutos y medio.
- 'Laboratory': 4 cámaras en las esquinas de una habitación, iluminación interior. 6 personas entrando en la habitación y moviéndose por ella, sin salir.
- 'Campus': 3 cámaras en la puerta de un edificio, iluminación exterior. Hasta 7 personas caminando enfrente de ella, entrando y saliendo de la escena.

Figura 4.1: Vistas de las cámaras en cada uno de los datasets utilizados: 'Terrace', 'Laboratory' y 'Campus' (de izquierda a derecha).

4.1.2. Métricas

Las métricas utilizadas para la evaluación del sistema comparando la salida de éste con el *ground truth* existente han sido MOTA y MOTP. Estas métricas son conocidas y utilizadas en la literatura para comparar distintos algoritmos de *tracking*[12]. Existen implementaciones de estas métricas disponibles en multitud de lenguajes, debido a su simplicidad se ha optado por realizar una implementación propia de ellas.

La **métrica MOTA**, ec. 4.1, da una indicación del número de errores que ha cometido el sistema, a modo de *recall*. Se calcula dividiendo el número total

de fallos (personas perdidas m, falsos positivos fp y personas mal identificadas mme) por el número total de elementos en el *ground truth* g,

$$MOTA = 1 - \frac{\sum_t (m(t) + fp(t) + mme(t))}{\sum_t g(t)}. \tag{4.1}$$

Un valor de 1 significa que no se ha cometido ningún error, mientras que un valor de 0 significa que se ha fallado en todas las situaciones.

La **métrica MOTP**, ec. 4.2, se calcula como la media entre todas las detecciones c de la distancia euclídea entre el punto medio de la región real proporcionada por el *groundtruth* con la devuelta por el sistema $dist_i$, medida en píxeles de la imagen. Proporciona una medida de precisión.

$$MOTP = \frac{\sum_{i,j} dist_i(t)}{\sum_t c(t)} \tag{4.2}$$

Un valor de 0 significa que todas las detecciones, por pocas que sean, han sido perfectas mientras que un valor mayor indica el error cometido, en media (por ejemplo, un valor de 10 significa que la distancia media entre las personas detectadas y las personas reales ha sido de 10 píxeles).

También se han diseñado visualizaciones y **otras métricas** nuevas, detalladas a continuación, creadas para analizar el resultado particular del sistema desarrollado. Hay que destacar que todas las métricas son mono-cámara, por lo que su evaluación se realiza para cada cámara por separado, mediando el resultado entre todas ellas cuando se indique.

1. Gráfica del mejor valor de IOU (*bestIOU*): Para cada frame y cada persona se representa una situación particular con un color diferente en función de si la persona se encuentra o no visible y si ha sido detectada correctamente o no (ver ejemplo en la Figura 4.2 izq). En el caso de que la persona se encuentre y haya sido detectada se muestra el valor de la función IOU entre ambas regiones, desde rojo-abajo para IOU=0 hasta verde-arriba para IOU=1. Esto permite observar de manera visual como de bueno ha sido el sistema detectando a las distintas personas, independientemente de la identificación asignada.

2. Gráfica de todas las combinaciones del IOU (*allIOU*): Para cada frame y cada combinación de personas detectadas con personas en el *ground truth*, se muestra el valor de la función IOU entre ambas regiones, o negro en caso de no estar presente (ver ejemplo en la Figura 4.2 der). En el eje de ordenadas se muestra cada persona detectada, que a su vez agrupa a cada persona del *ground truth*. Para facilitar la visualización cada una de las personas del *ground truth* se ha representado con un color diferente con transparencia igual al IOU. Esta gráfica permite visualizar el tiempo máximo que el sistema ha estado siguiendo a una misma persona, cuándo ha pasado a detectar otra diferente o si ha habido dos detecciones equivalentes, entre otros.

3. Duración máxima de seguimiento: MAXDUR. Tal y como se especifica para las métricas MOTA y MOTP se asume que un valor de IOU mayor

que 0.5 corresponde a una detección correcta. Con este valor se pretende comprobar cual es la duración máxima que una cámara ha logrado seguir a una persona, en número de frames.

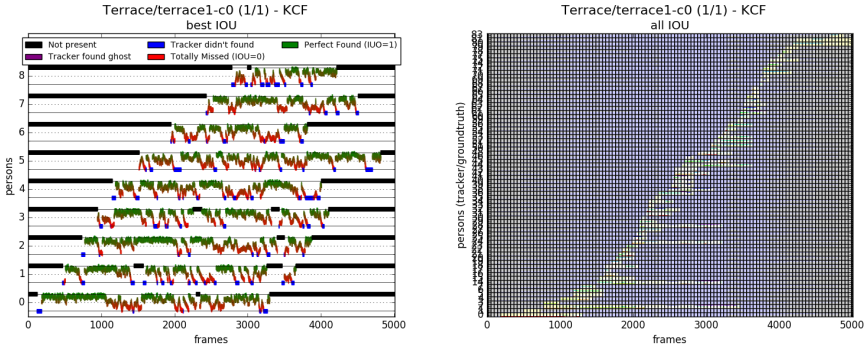

Figura 4.2: Ejemplos de gráficas *bestIOU* (izq) y *allIOU* (der)

4.2. Evaluación de módulos individuales

El sistema propuesto requiere el uso de dos módulos externos, detector y un algoritmo de seguimiento en una única cámara, que llamaremos *tracker*, para los que se evaluaron distintas alternativas disponibles.

4.2.1. Detector

El sistema requiere de un módulo 'detector' capaz de devolver una región (rectángulo) para cada una de las personas en una imagen dada. Debido a que la librería OpenCV contiene varias implementaciones de detectores se probó a utilizarlos directamente. Sin embargo una vez ejecutado se comprobó que eran demasiado erróneos, pues visualmente se veía con claridad que no detectaban la mayoría de personas y muchas de las detecciones eran de elementos de la escena. Por lo tanto, se descartó su uso.

Otra opción estudiada es utilizar un detector de objetos que incluya la clase *persona*, por ejemplo el Mask-RCNN [4] basado en deep learning. Este detector ha demostrado devolver muy buenos resultados en escenas muy variadas, y tras su ejecución se comprobó que efectivamente los resultados proporcionados para el dataset usado en este trabajo eran casi perfectos. Todos los resultados que se describen en este trabajo han sido utilizando este detector.

4.2.2. *Tracker*

El otro módulo necesario consiste en un módulo de seguimiento, *tracker*, para una cámara. Es decir, un algoritmo que a partir de una región de inicialización de donde está una persona en una imagen, va estimando su posición en todas las imágenes posteriores. Al igual que con el detector, la librería OpenCV proporciona varios trackers, que se lanzaron y analizaron para ver sus resultados. En concreto se probaron los siguientes: 'BOOSTING', 'MIL', 'KCF', 'TLD' y

'MEDIANFLOW'. De ellos se tuvieron que descartar el 'MIL', 'TLD' y 'MEDIANFLOW' por su malo desempeño, pues visualmente se observó que fallaba mucho y no era capaz de seguir correctamente a las personas. 'KCF' se observó que no destacaba, pero funcionaba muy rápido sin demasiados fallos, y 'BOOSTING' daba los mejores resultados de todos ellos, aunque el tiempo de cálculo se incrementaba notablemente respecto al resto. Además de la evaluación subjetiva, se contrastó la información obtenida con otras fuentes [13] y [14] que también llegaban a la misma conclusión.

Al realizar los experimentos con las secuencias completas, 'BOOSTING' tuvo que ser descartado pues el uso de memoria sobrepasaba el disponible bloqueando el ordenador sin dejar que el sistema concluyera satisfactoriamente. Por esta razón todos los experimentos han sido realizados utilizando el *tracker* 'KCF' disponible en la librería OpenCV.

4.2.3. Influencia de la frecuencia de lanzamiento del detector

El sistema se diseñó para utilizar la información devuelta por el detector únicamente cuando ésta se encuentra disponible. Esto es debido a que normalmente el tiempo de cálculo del detector es superior al del resto de componentes, por lo que es mejor lanzarlo cuantas menos ocasiones mejor.

Para comprobar cada cuanto es aceptable lanzar el detector se evaluaron las métricas MOTA y MOTP con la ejecución del sistema en las distintas escenas multicámara lanzando el detector cada X frames con las siguientes variaciones: X=1 (el detector se lanza en todos los frames), X=5 (el detector se lanza cada 5 frames) y X=10 (el detector se lanza cada 10 frames). En las tablas 4.1 y 4.2 se muestran los resultados obtenidos.

MOTA	X=1	X=5	X=10
Terrace	0.72	**0.73**	0.69
Laboratory	0.70	**0.71**	0.67
Campus	**0.56**	0.46	0.37

Tabla 4.1: Evaluación del seguimiento en una cámara lanzando el detector cada X frames. Para cada escena se muestra la media de MOTA obtenida por cada cámara.

MOTP	X=1	X=5	X=10
Terrace	11.83	**11.65**	12.90
Laboratory	17.56	**16.60**	18.36
Campus	**11.72**	14.19	12.02

Tabla 4.2: Evaluación del seguimiento en una cámara lanzando el detector cada X frames. Para cada escena se muestra la media de MOTP obtenida por cada cámara.

En estos resultados se observa que en las escenas *Terrace* y *Laboratory* la configuración que da mejores resultados corresponde a ejecutar el detector cada 5 frames. Sin embargo en la escena *Campus* el mejor resultado es ejecutando el detector en cada frame. Esto es debido a que en esta última escena las personas

entran y salen del campo de visión sin casi cruzarse, mientras que en las otras una vez que entran se mantienen visibles al menos en una cámara aumentando el número de cruces. Como el objetivo del algoritmo era centrarse en las situaciones con cruces, y además ejecutar el detector en cada frame conlleva un mayor tiempo de cálculo, se concluye que el valor de X=5 es adecuado y por tanto en el resto de experimentos el detector se ejecuta siempre cada 5 frames.

4.3. Evaluación del sistema completo

En esta sección se evalúan los experimentos consistentes en ejecutar el sistema para una o varias cámaras, comprobando la mejora que se obtiene. Las tablas 4.3, 4.4 y 4.5 muestran los resultados obtenidos. El modo de operar es el mismo para todas las escenas, en el que se comparan las siguientes dos ejecuciones:

1. Se ejecuta el sistema para cada cámara por separado, como si se tuviese una única cámara disponible. Esto corresponde a no utilizar varias cámaras, a modo de base. Se obtienen N valores para cada métrica (N el número de cámaras en la escena) y se median.

2. Se ejecuta el sistema para las N cámaras disponibles, por lo que el sistema ha tenido acceso a todas las cámaras y por tanto la información ha sido consensuada y evaluada en conjunto. Esto también proporciona N valores para cada métrica, que también se median. Esta situación corresponde a la mejora esperada por utilizar varias cámaras en lugar de una sola.

MOTA	1 cámara	N cámaras
Terrace (N=4)	0.60	**0.73**
Laboratory (N=4)	0.58	**0.71**
Campus (N=3)	0.45	**0.46**

Tabla 4.3: Evaluación del seguimiento lanzando el algoritmo con 1 o varias cámaras. Para cada escena se muestra la media de MOTA obtenida por cada cámara.

MOTP	1 cámara	N cámaras
Terrace (N=4)	15.38	**11.65**
Laboratory (N=4)	22.43	**16.60**
Campus (N=3)	14.64	**14.19**

Tabla 4.4: Evaluación del seguimiento lanzando el algoritmo con 1 o varias cámaras. Para cada escena se muestra la media de MOTP obtenida por cada cámara.

MAXDUR	1 cámara	N cámaras
Terrace (N=4)	325.97	**335.11**
Laboratory (N=4)	223.92	**234.29**
Campus (N=3)	164.77	**171.43**

Tabla 4.5: Evaluación del seguimiento lanzando el algoritmo con 1 o varias cámaras. Para cada escena se muestra la media de MAXDUR obtenida por cada cámara.

En las tres tablas (4.3, 4.4 y 4.5) se observa que la ejecución con múltiples cámaras mejora los resultados en todas las escenas, lo que indica que el sistema es capaz de cumplir su objetivo y mejorar la detección de múltiples personas mediante el uso de múltiples cámaras.

También se puede observar que *Campus* proporciona los peores resultados en general, pues en esta escena las personas dejan de ser visibles por todas las cámaras en varias ocasiones. Este hecho se observa con más claridad si comparamos las gráficas *bestIOU* y *allIOU*. A continuación se mostrarán algunas de estas visualizaciones más representativas para entender los resultados obtenidos. En el anexo A se adjuntan todas ellas.

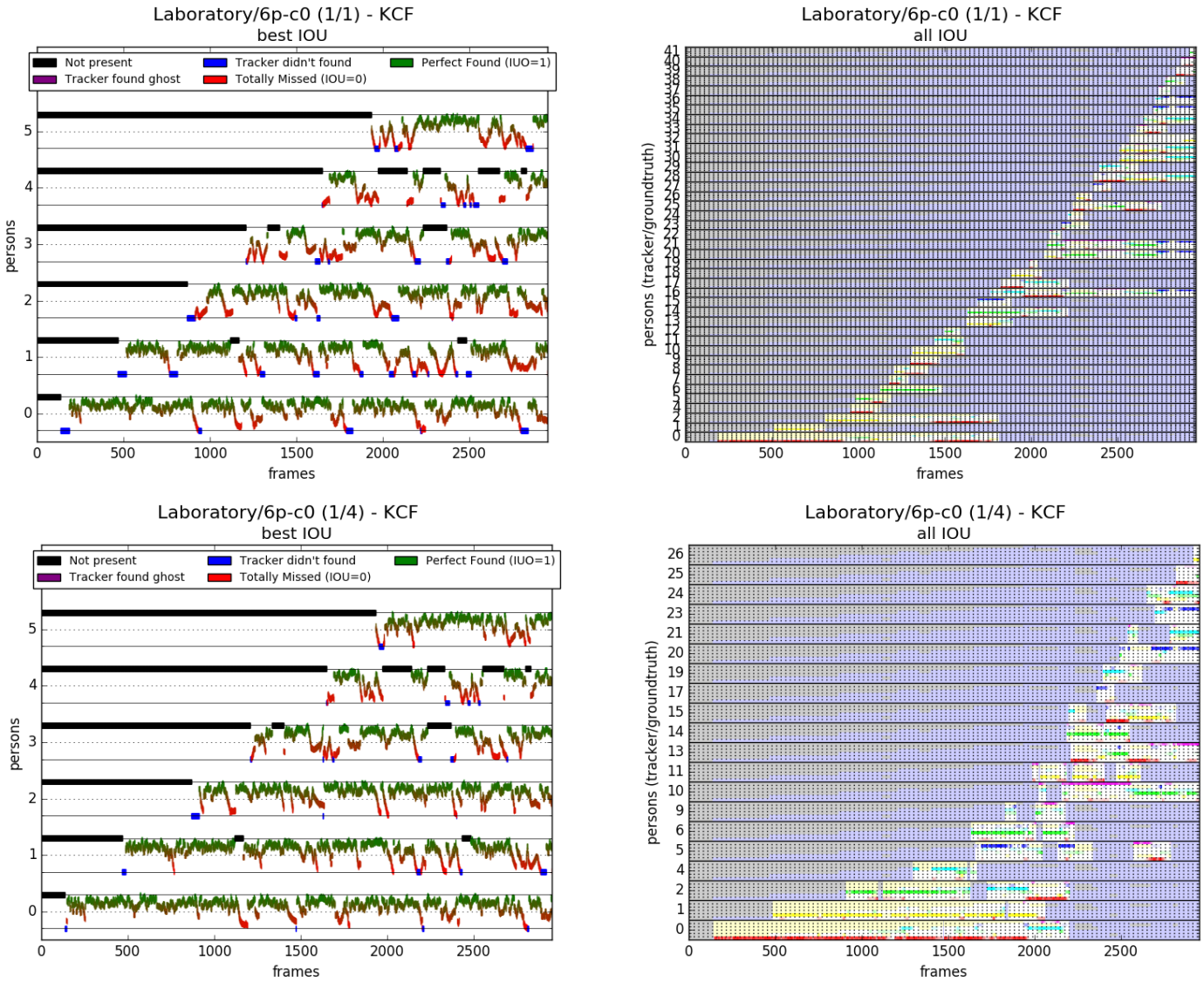

Figura 4.3: Comparativa 1 vs 4 cámaras: *Laboratory-c0*

La Figura 4.3 muestra la comparativa de una de las cámaras de la escena *Laboratory* respecto a ejecutar el sistema con ella únicamente (fila superior) o con las 4 disponibles (fila inferior). En la primera columna se representa para cada persona del *groundtruth* (persona real, eje Y) la mejor evaluación del sistema en cada frame (eje X), medida en el valor devuelto por la función IOU. Se observa que aunque parecen similares, la ejecución con solo una cámara (gráfica superior) contiene más frames en los que el tracker ha fallado o no ha encontrado a la persona (puntos azules y rojos) mientras que en la ejecución con las 4 cámaras (gráfica inferior) muchos de estos 'huecos' se han rellenado, razón por la que la longitud media aumenta en casi 10 frames, tal y como mostraba la tabla 4.5. En la segunda columna se muestra la relación entre cada persona

detectada (eje Y) con cada persona del *groundtruth* (colores). Se puede ver que las dos primeras personas que aparecen, mostradas en rojo y amarillo, han sido detectadas correctamente casi toda la primera mitad del vídeo con los identificadores 0 y 1 respectivamente, hasta llegar a un punto en el que saltan a ser identificadas como 2 y 11, y posteriormente otros. La tercera y cuarta persona (amarillo y azul) también comienzan correctamente hasta que 'saltan' a otros identificadores, mientras que la quinta y sexta (morado y violeta) al aparecer cuando ya había mucha gente en la escena no son seguidas durante un gran número de frames por el mismo identificador. Es notable destacar que el sistema empieza a fallar cuando en la escena se encuentran 5 o más personas, lo que puede indicar que al haber más personas que cámaras se producen situaciones en las que una persona se oculta en todas ellas simultáneamente, lo que facilita que se pierda.

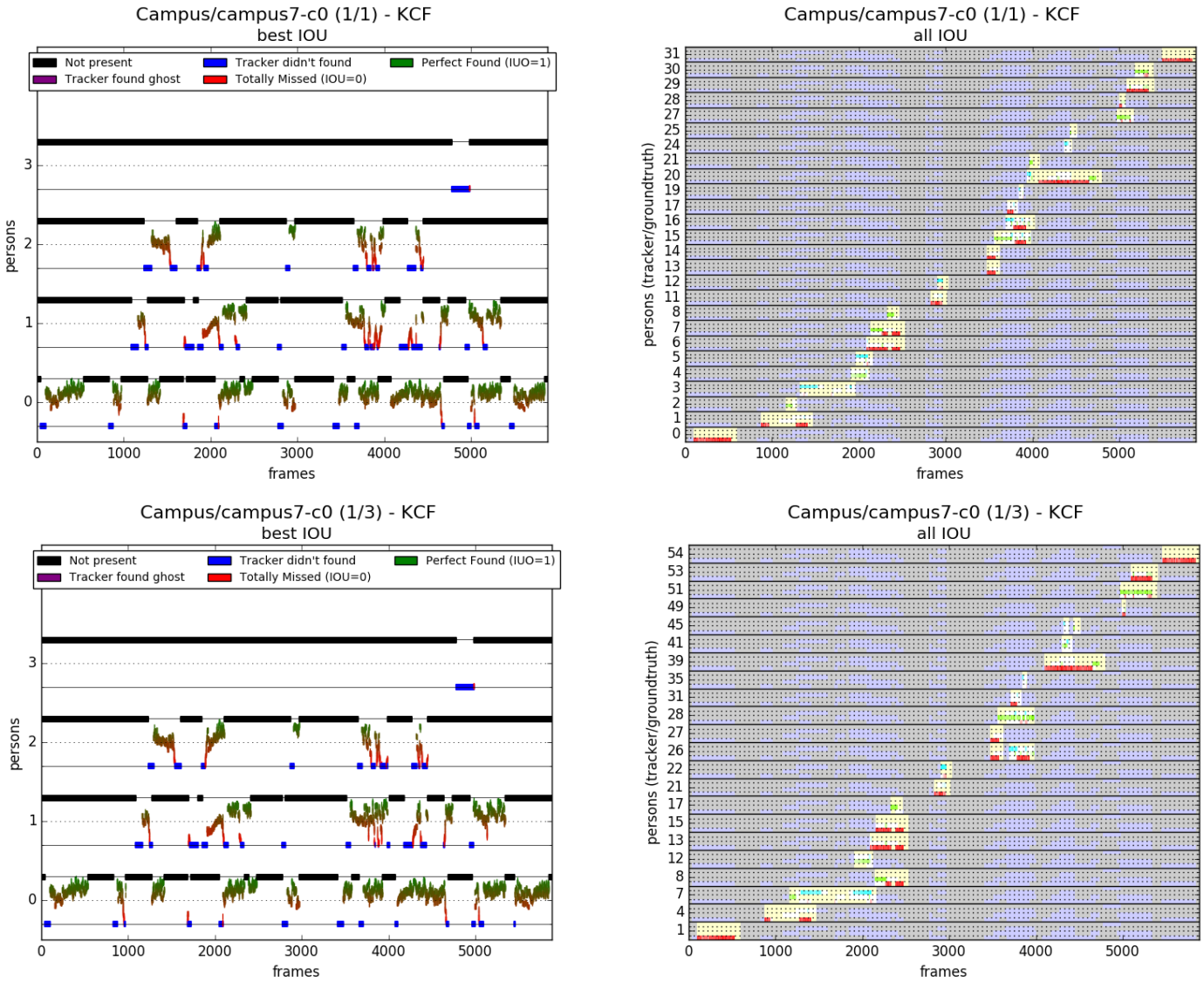

Figura 4.4: Comparativa 1 vs 3 cámaras: *Campus-c0*

En la comparativa de la Figura 4.4 se muestra la misma información para la escena *Campus*, cuyos resultados mostrados en las tablas previas habían sido de los peores. Se puede comprobar que a diferencia del *Laboratory* en donde las personas entraban y se mantenían en la escena (sólo había lineas negras, persona no visible, al principio del gráfico y un poco a mitad) en esta escena sin embargo las personas están ocultas durante gran parte del vídeo, la última de ellas incluso sólo se muestra durante unos pocos frames al final (y no es detectada). Es probablemente ésta la razón del mal resultado obtenido con la escena, pues aquí la dificultad se encuentra en detectar correctamente a las personas cuando aparecen, no en seguirlas, ya que desaparecen poco después. Esta necesidad de detección temprana explica también que en el experimento

del tiempo del detector (tablas 4.2 y 4.1) lanzar más veces el detector mejorase los resultados, pues las personas eran detectadas antes.

Capítulo 5

Conclusiones y Trabajo Futuro

En este proyecto se ha diseñado y evaluado un sistema para seguimiento de múltiples personas desde múltiples cámaras. Se ha presentado la formulación de un algoritmo para mejorar la detección gracias al uso de múltiples cámaras. También se ha desarrollado una implementación del mismo, la cual se ha evaluado utilizando datasets públicos para este problema.

En relación a los objetivos del trabajo, todos ellos se han alcanzado. Se ha realizado un estudio del estado del arte en el campo de la visión por computador, y se han comparado alternativas de detección y seguimiento mono-cámara. A partir de estas bases se han propuesto mejoras y se ha desarrollado un nuevo método multicámara, trabajando con datasets diseñados para este problema.

A continuación se discutirán las conclusiones más detalladas del mismo y los posibles trabajos futuros.

5.1. Conclusión

Los experimentos realizados muestran que el algoritmo propuesto es capaz de mejorar el seguimiento de múltiples personas en las escenas gracias al uso de múltiples cámaras, respecto a utilizar únicamente una de ellas. En particular, obtiene resultados más robustos cuando las personas se cruzan en la escena de forma que en al menos una cámara no se produce cruce, o cuando una se oculta pero reaparece poco después. Es el hecho de utilizar varias cámaras el que permite que mientras en una de ellas las personas sean distinguibles esa información se propague al resto.

Los casos más débiles son cuando las personas permanecen poco tiempo visibles entre todas las cámaras, pues al algoritmo no le da tiempo a detectar que es una nueva persona y de identificarla correctamente. Tampoco es capaz de recuperarse correctamente de situaciones en las que debido a la existencia de muchas personas simultáneamente (número de cámaras+1) se dan situaciones en las que una misma persona permanece oculta en todas las cámaras a la vez, por lo que se pierde.

5.2. Trabajo Futuro

Los módulos externos utilizados, el *tracker* de una cámara y el detector de personas, son críticos del sistema pues en ellos depende el correcto funcionamiento. Como trabajo futuro se plantea la programación interna de estos módulos utilizados, especialmente del tracker, pues ésta permitiría un mayor control sobre el método utilizado y las variables internas que almacena, en lugar de depender de una implementación dada. Otras de las modificaciones que se plantean consisten en cambiar las regiones devueltas por el detector una vez que la información ha sido consensuada entre las cámaras, decidir cuando es mejor lanzar el detector en lugar de hacerlo periódicamente, o incluso en que zona de la imagen es conveniente hacerlo, así como realizar una ejecución con otros datasets, tanto existentes como propios.

Respecto a la identificación de las personas, uno de los problemas que se encontraron es el alto número de personas diferentes devueltas por el algoritmo, especialmente en las escenas en las que con frecuencia las personas salían de ella (dejaban de ser visibles en todas las cámaras) por lo que tras su reidentificación se le asignaba un nuevo identificador. Utilizar algún método que permita obtener similitudes entre las detecciones, por ejemplo con el color predominante de la detección o con algún rasgo característico de ésta, ayudaría no solo a reducir el número de identificadores totales utilizados y por tanto de personas únicas detectadas, sino que ayudaría a la velocidad de seguimiento e identificación general.

Apéndice A

Detalles de resultados de la evaluación

En este anexo se muestran todas las gráficas obtenidas al ejecutar el experimento descrito en la sección 4.3. Cada figura muestra los resultados de una cámara de cada uno de los datasets utilizado, en total $4 + 4 + 3 = 11$ figuras. Dentro de cada una se muestran 4 gráficas, las dos superiores correspondientes a la ejecución del sistema utilizando únicamente la cámara en cuestión, y debajo los resultados tras la ejecución con todas las cámaras disponibles (3 ó 4 según corresponda). En cuanto al tipo de gráficas cada una representa lo siguiente:

1. Gráficas de la izquierda: Para cada frame y cada persona se representa una situación particular con un color diferente en función de si la persona se encuentra o no visible y si ha sido detectada correctamente o no. En el caso de que la persona se encuentre y haya sido detectada se muestra el valor de la función IOU entre ambas regiones, desde rojo-abajo para IOU=0 hasta verde-arriba para IOU=1. Esto permite observar de manera visual como de bueno ha sido el sistema detectando a las distintas personas, independientemente de la identificación asignada.

2. Gráficas de la derecha: Para cada frame y cada combinación de personas detectadas con personas en el *groundtruth*, se muestra el valor de la función IOU entre ambas regiones, o negro en caso de no estar presente. En el eje Y se muestran cada persona detectada, que a su vez agrupa a cada persona del *groundtruth*. Para facilitar la visualización cada una de las personas del *groundtruth* se ha representado con un color diferente con transparencia igual al IOU. Esta gráfica permite visualizar el tiempo máximo que el sistema ha estado siguiendo a una misma persona, cuándo ha pasado a detectar otra diferente o si ha habido dos detecciones equivalentes, entre otros.

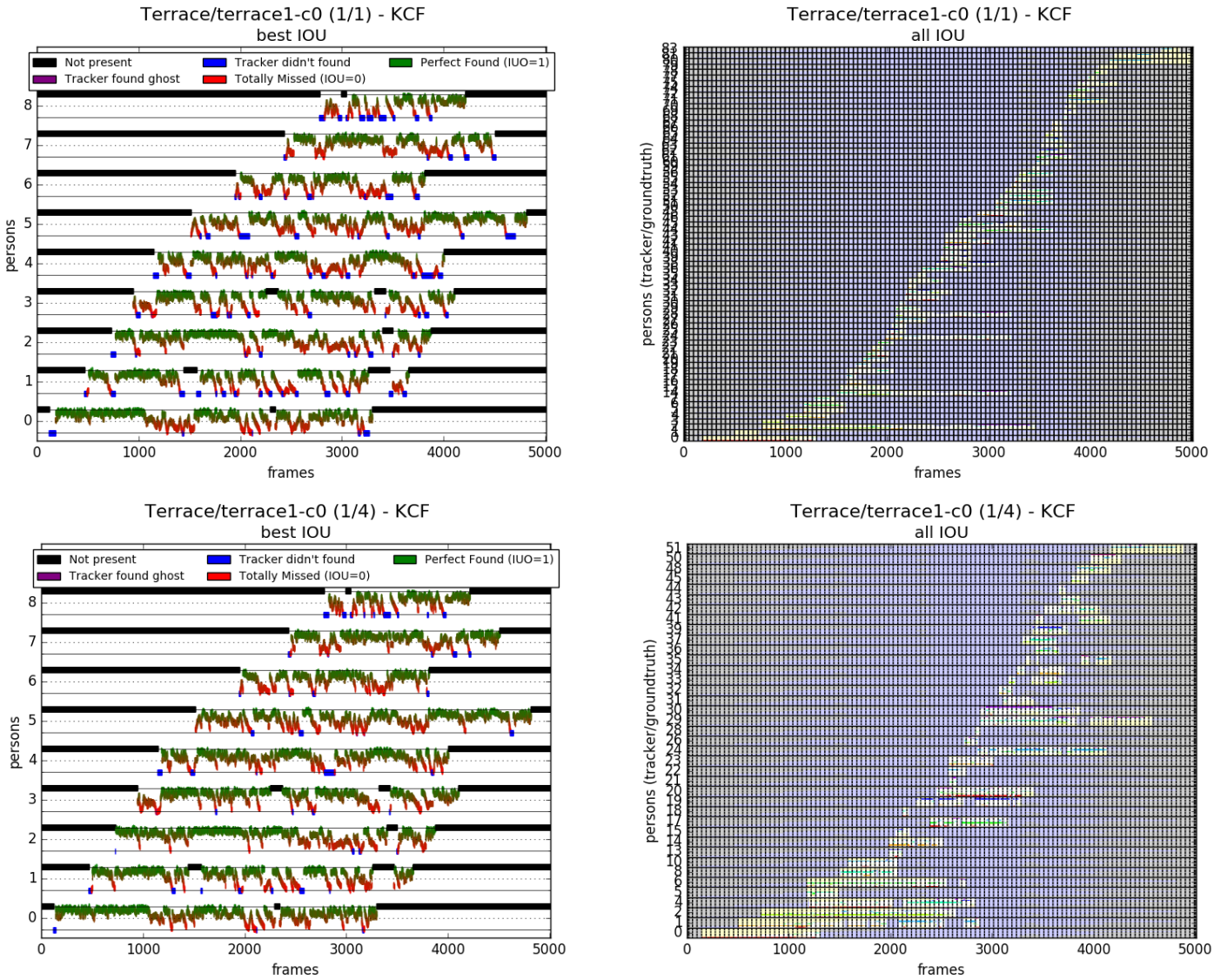

Figura A.1: Comparativa 1 vs 4 cámaras: *Terrace-c0*

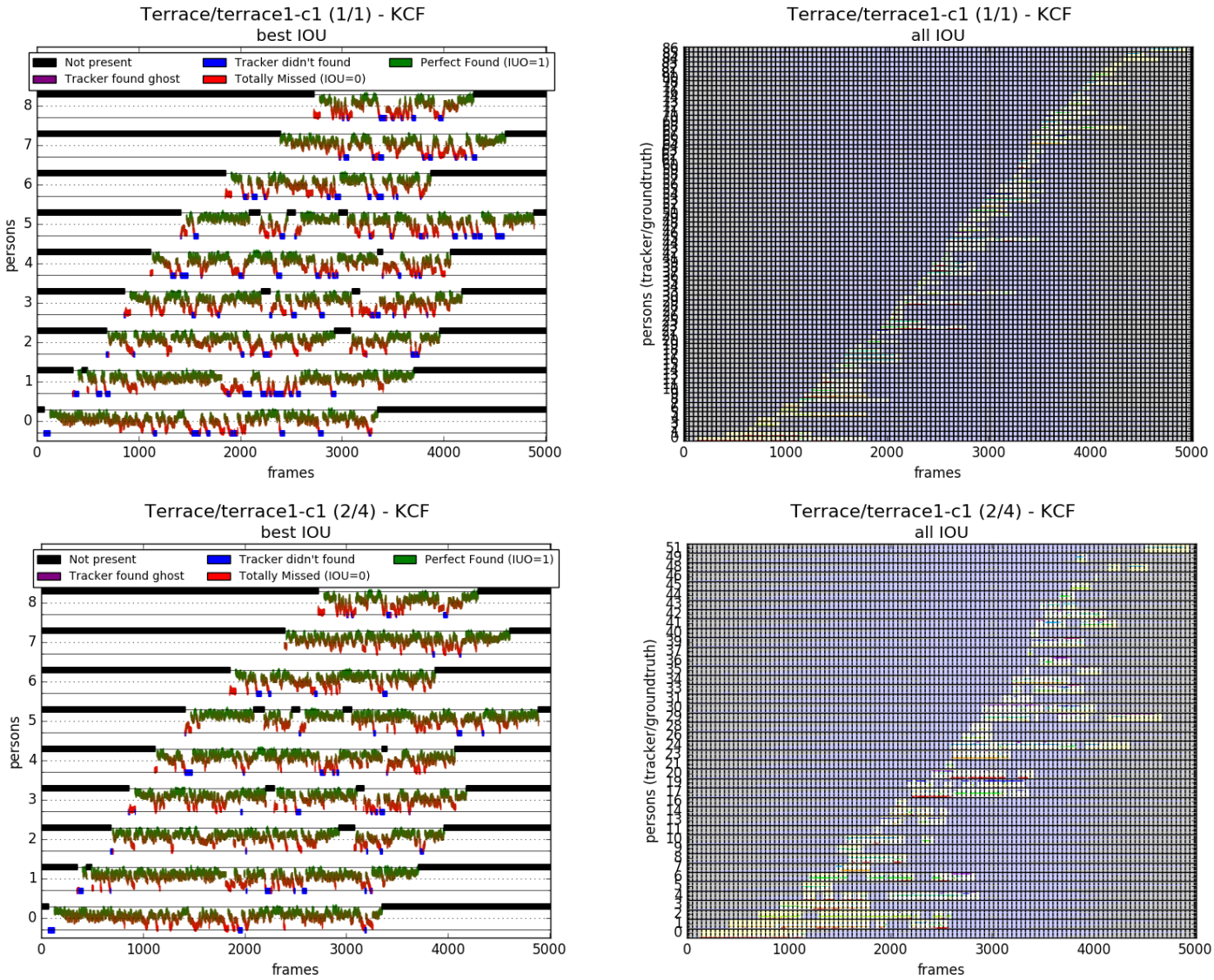

Figura A.2: Comparativa 1 vs 4 cámaras: *Terrace-c1*

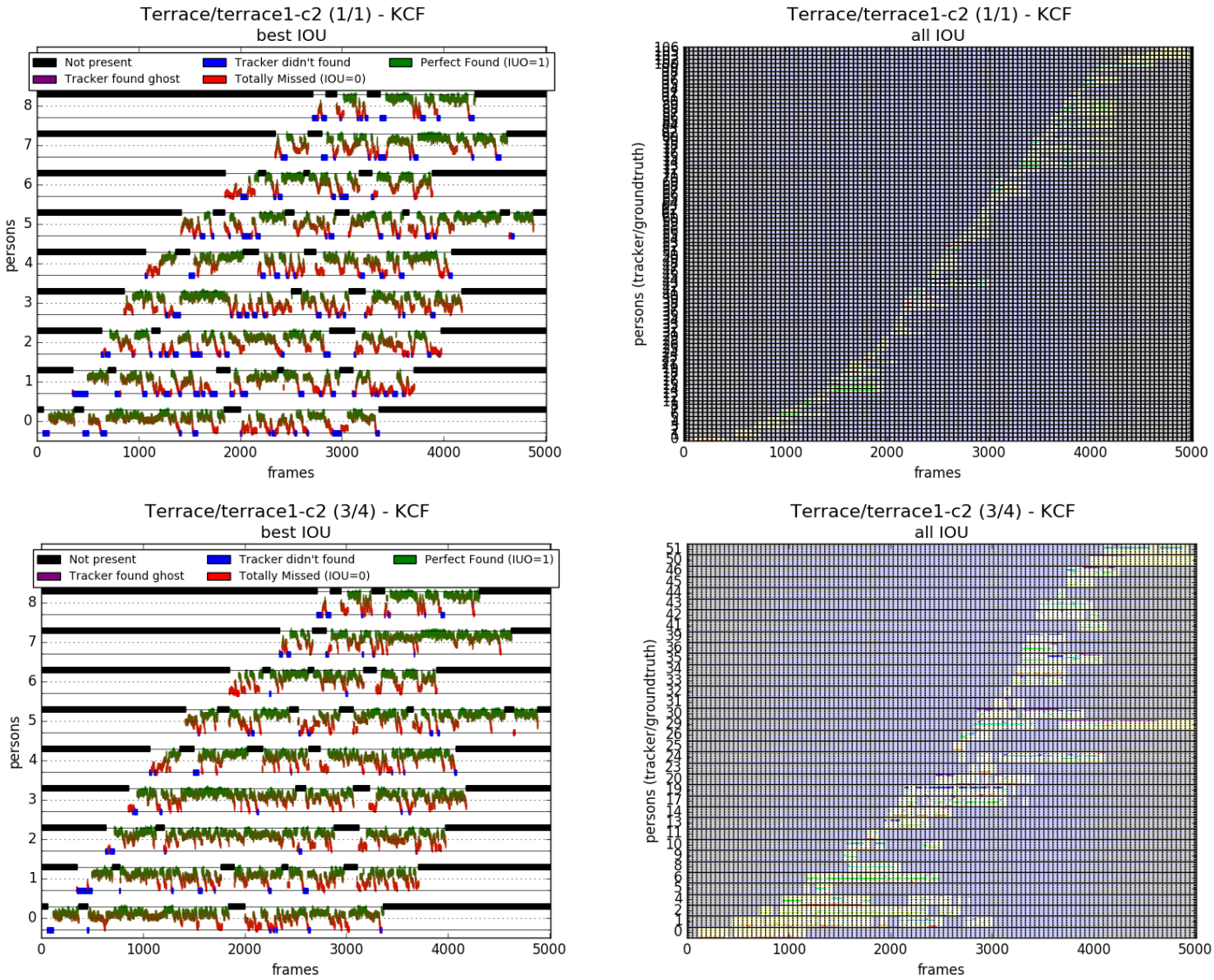

Figura A.3: Comparativa 1 vs 4 cámaras: *Terrace-c2*

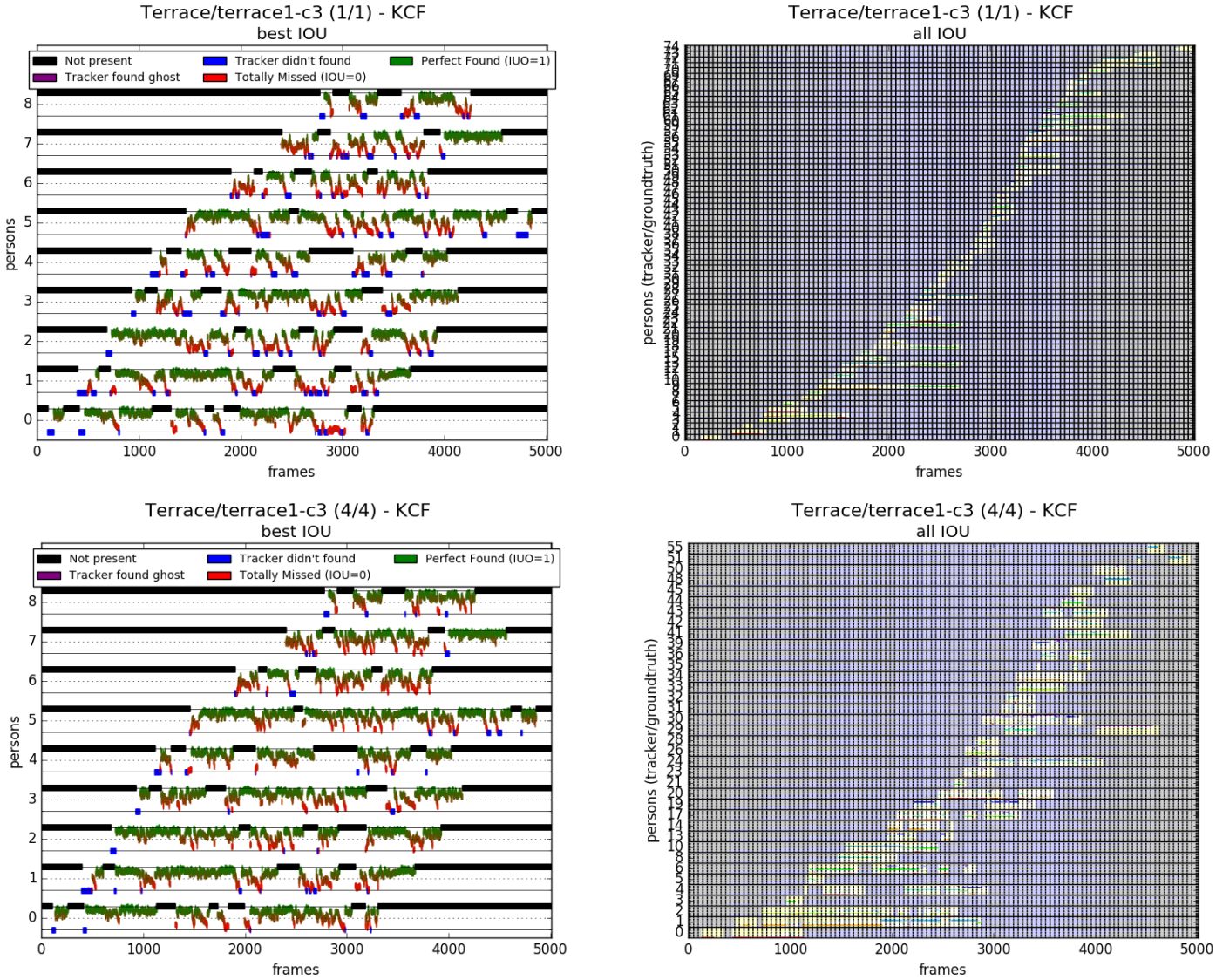

Figura A.4: Comparativa 1 vs 4 cámaras: *Terrace-c3*

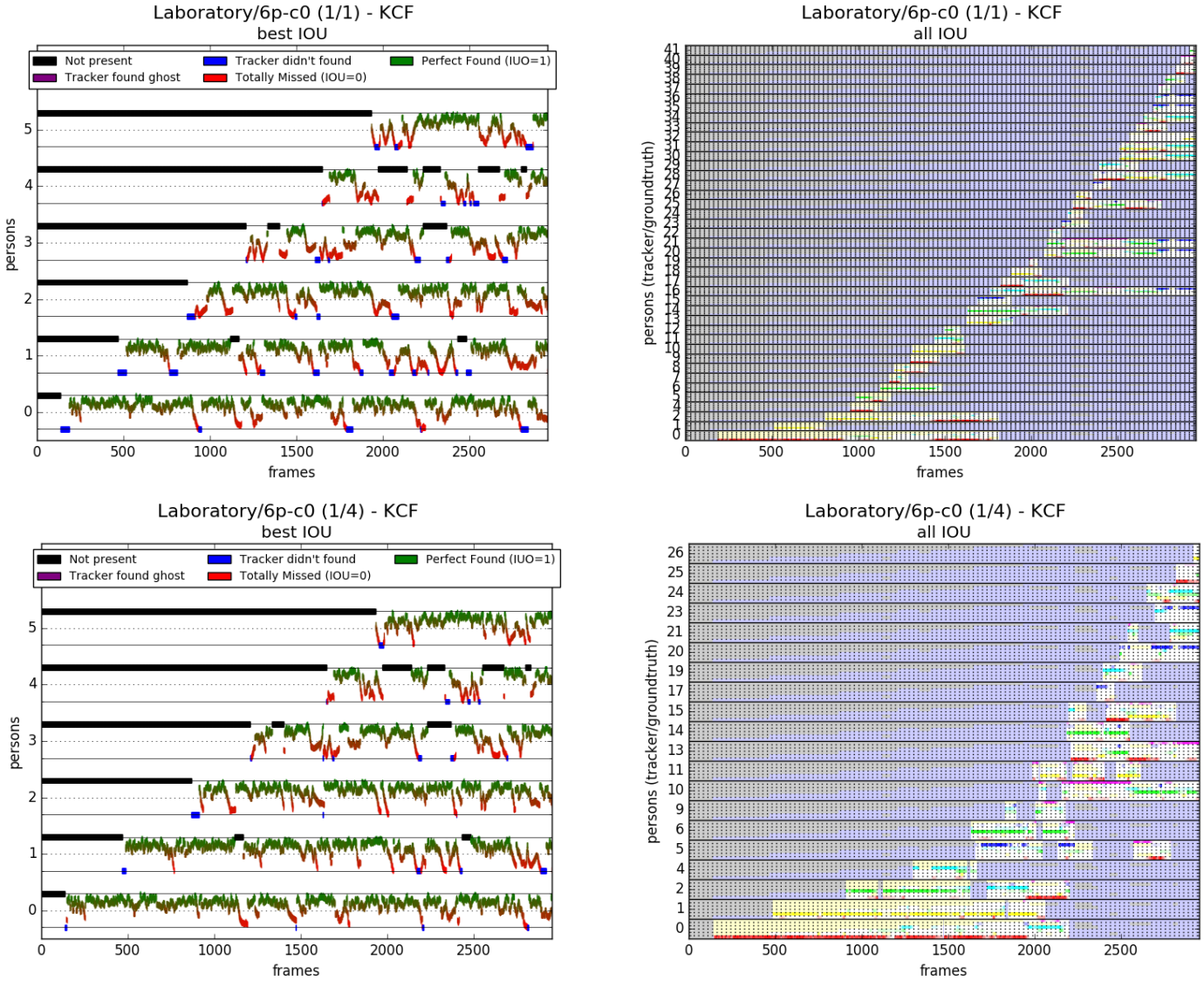

Figura A.5: Comparativa 1 vs 4 cámaras: *Laboratory-c0*

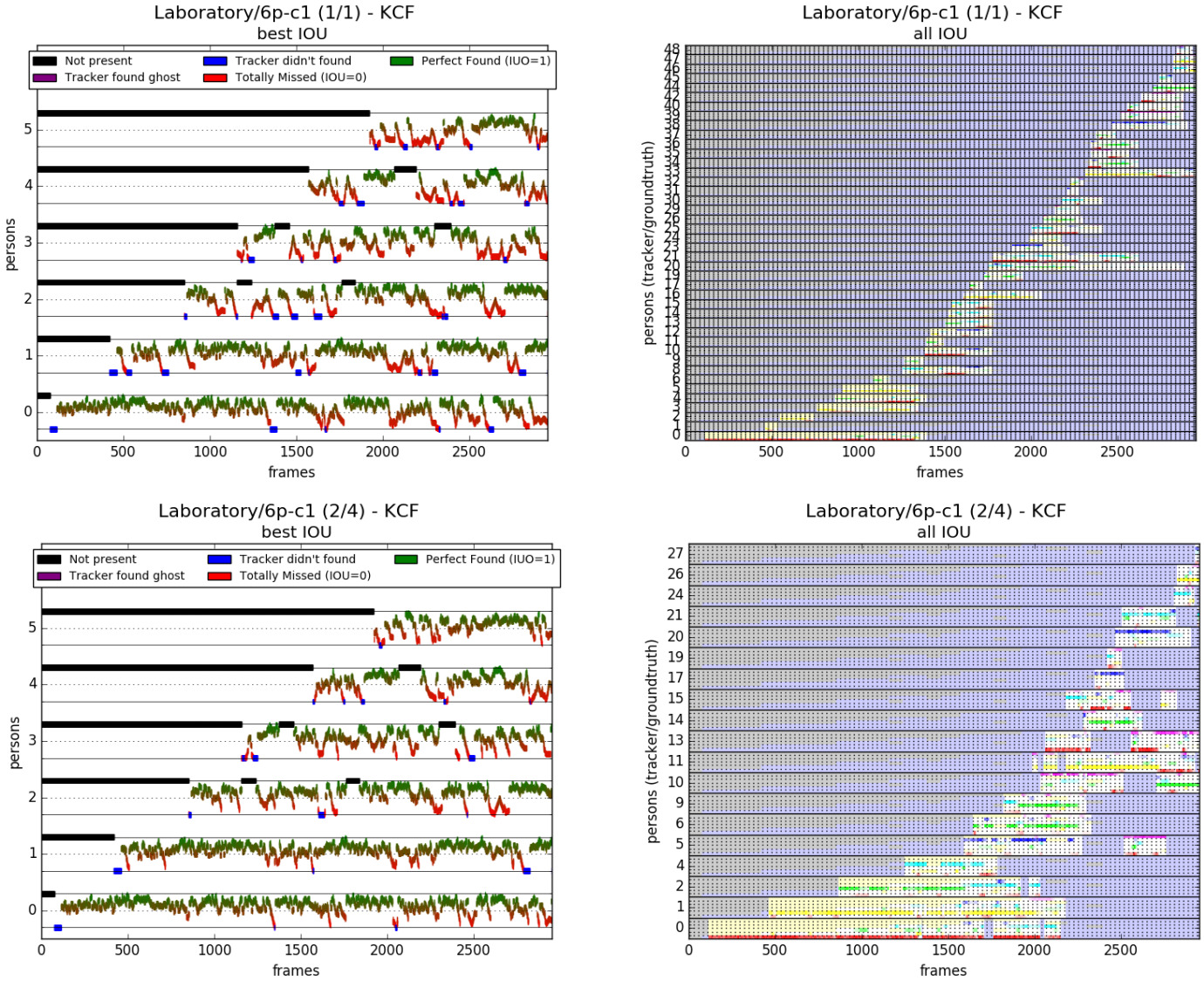

Figura A.6: Comparativa 1 vs 4 cámaras: *Laboratory-c1*

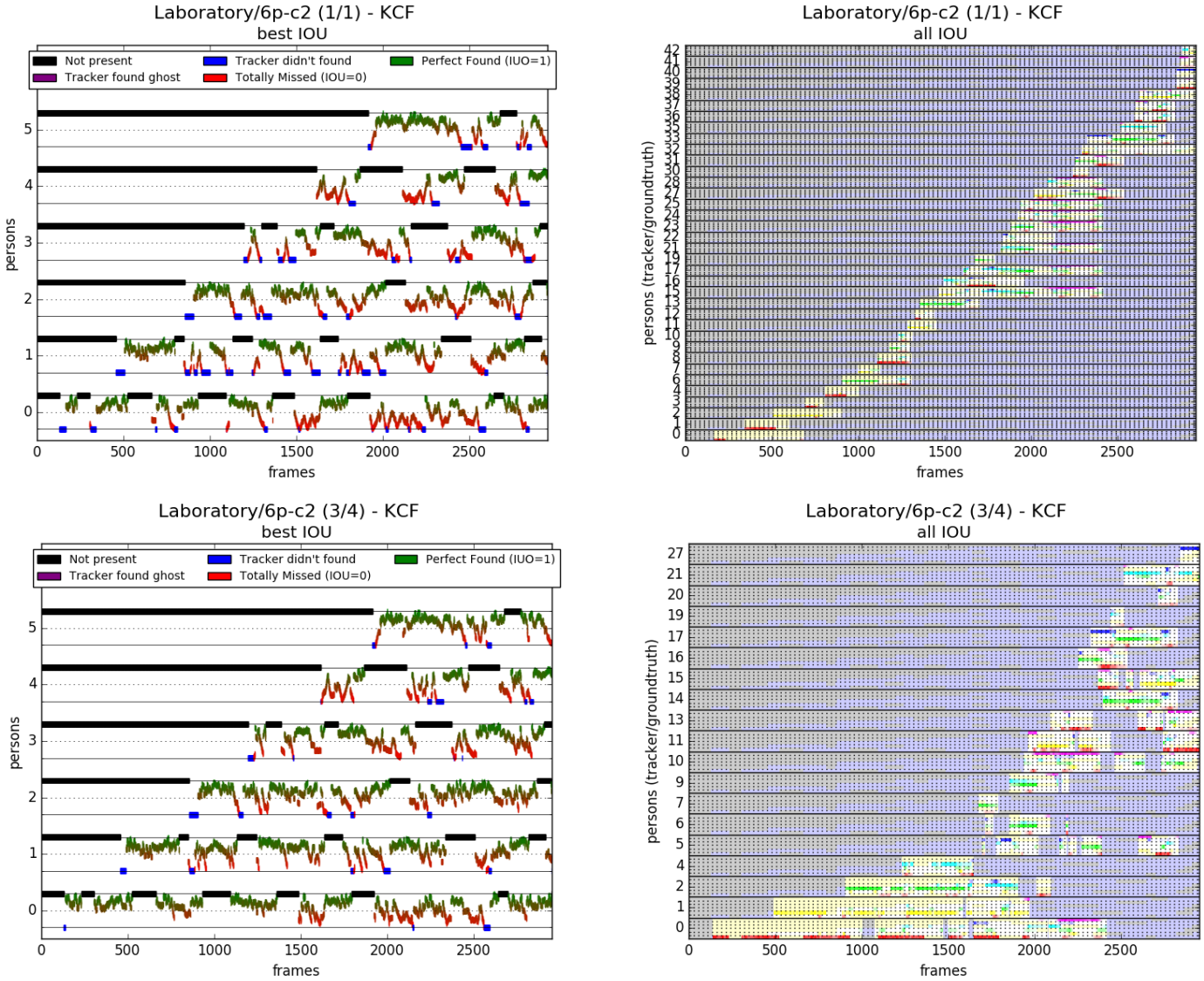

Figura A.7: Comparativa 1 vs 4 cámaras: *Laboratory-c2*

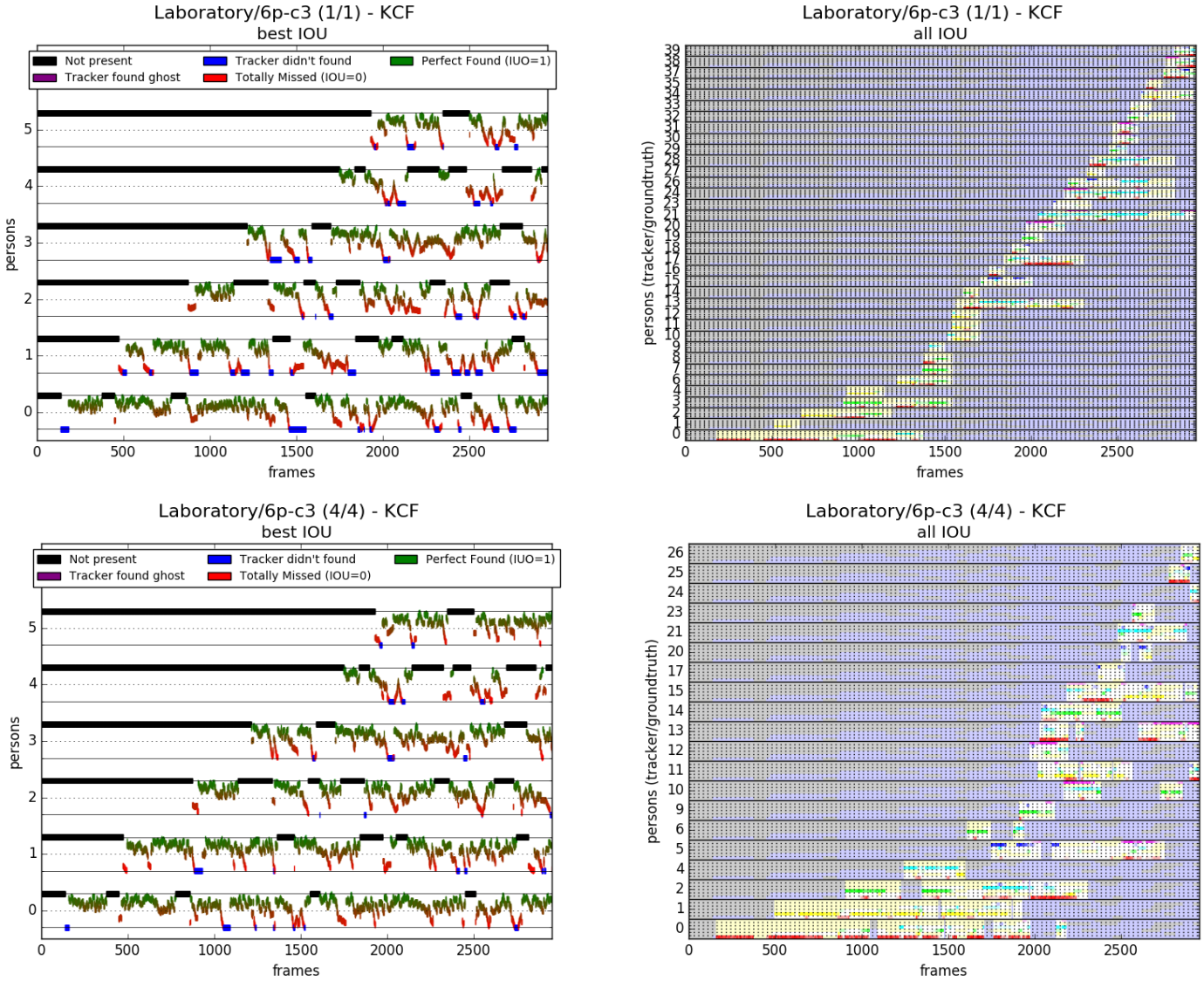

Figura A.8: Comparativa 1 vs 4 cámaras: *Laboratory-c3*

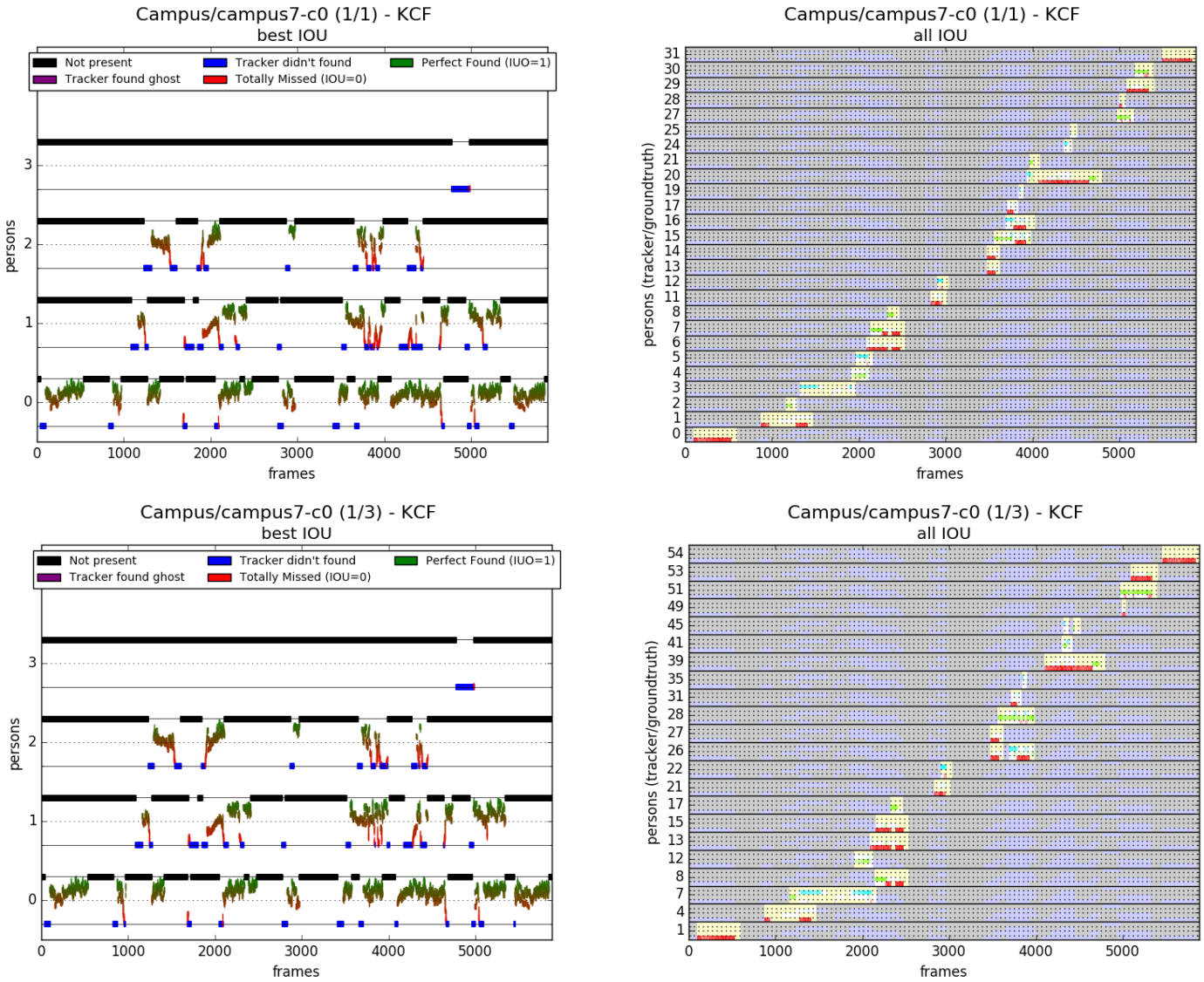

Figura A.9: Comparativa 1 vs 3 cámaras: *Campus-c0*

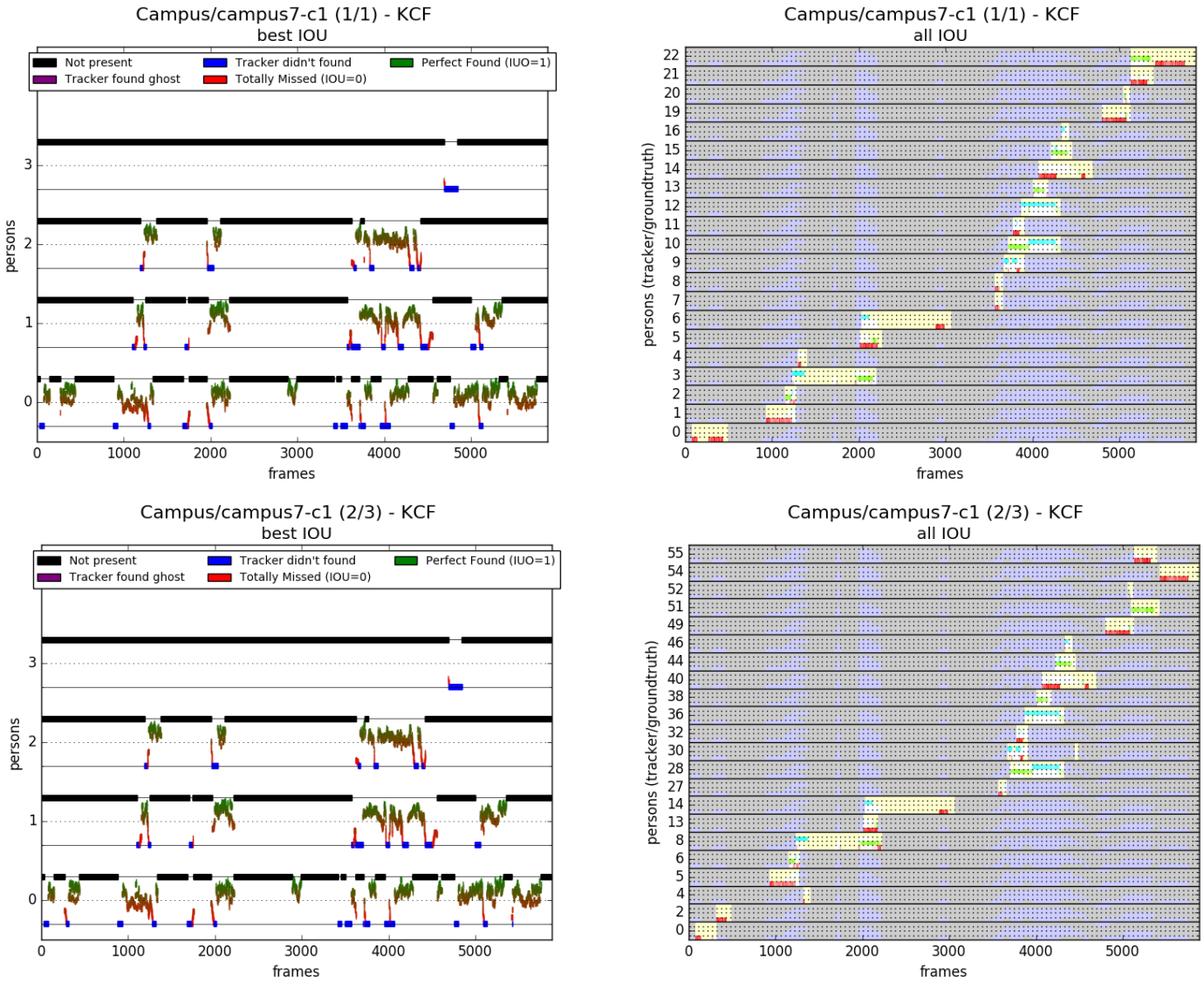

Figura A.10: Comparativa 1 vs 3 cámaras: *Campus-c1*

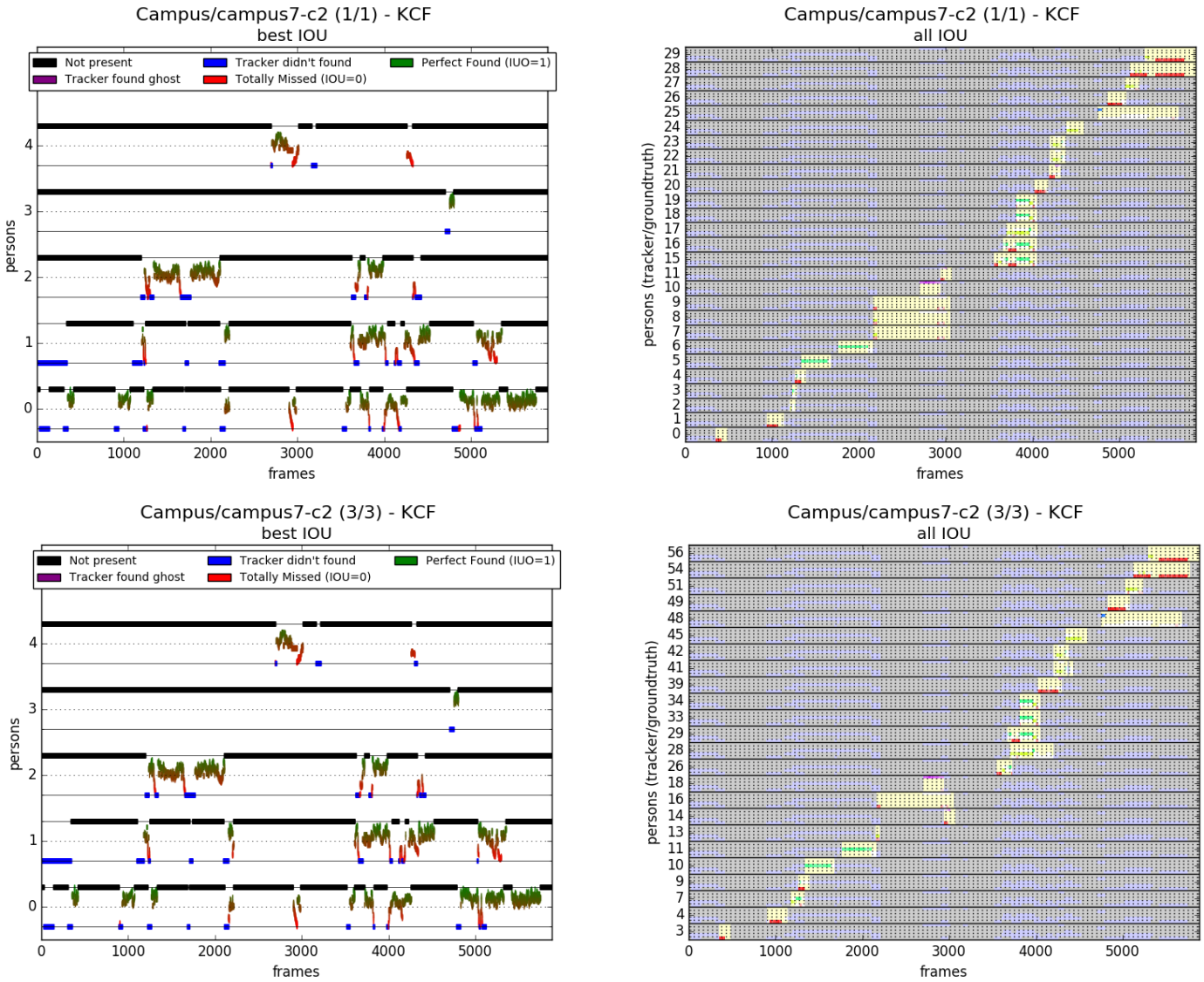

Figura A.11: Comparativa 1 vs 3 cámaras: *Campus-c2*

Apéndice B

Puesta en marcha del sistema completo

En este capítulo se explican los pasos necesarios para la instalación y ejecución del sistema, así como los programas y versiones necesarios para ello.

El código se encuentra disponible en `https://github.com/anacmurillo/multicam_unizar` Al consistir en scripts de python la ejecución del programa, así como algunos de los submódulos, se realiza lanzando el correspondiente fichero a través de un intérprete python. El algoritmo se encuentra en el fichero *multiCameraTrackerV2.py*, y su evaluación se realiza desde el fichero *metric_evaluation.py*. En la ejecución se puede ir observando el estado de las detecciones en cada cámara, así como de sus posiciones en el suelo (Figuras 3.2, B.1 y B.2).

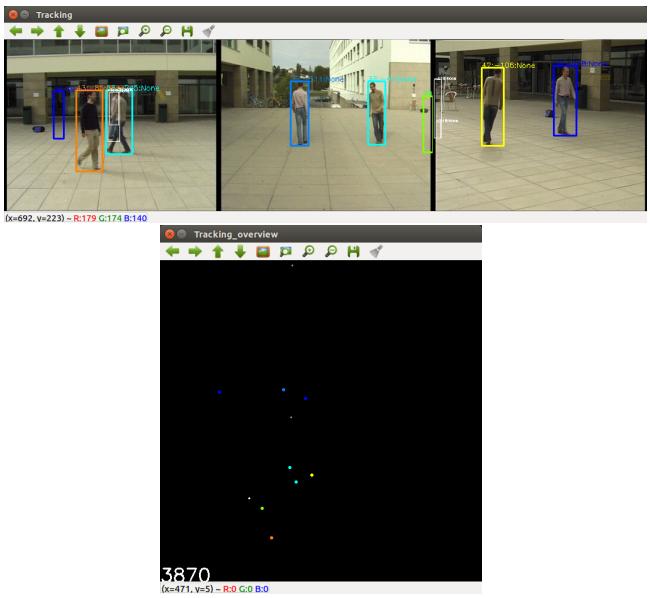

Figura B.1: Ventanas del sistema en ejecución para el dataset *Campus*. Arriba: Vista de las detecciones (regiones) de las personas en las distintas cámaras. Abajo: La posición (puntos) de las personas proyectada en el plano del suelo.

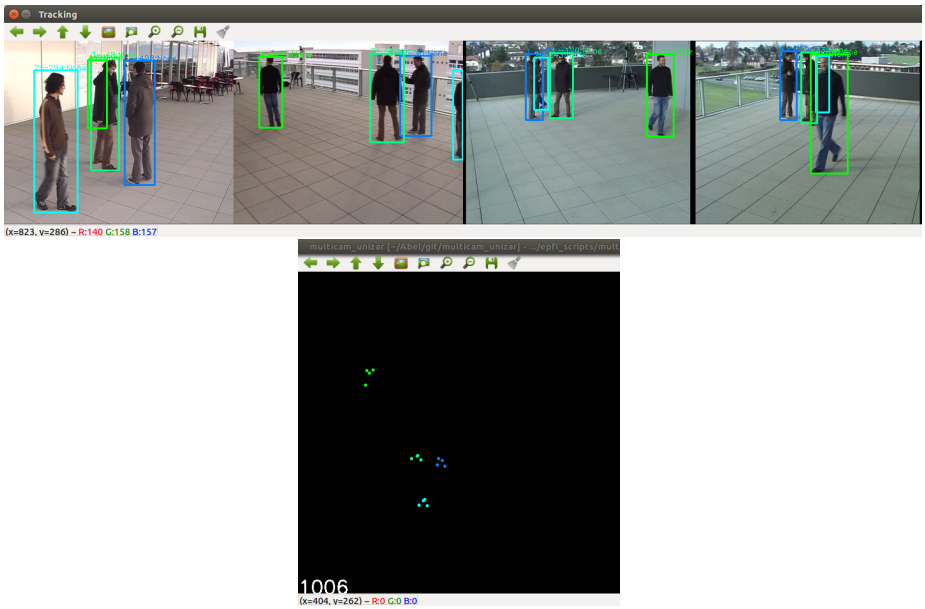

Figura B.2: Ventanas del sistema en ejecución para el dataset *Terrace*. Arriba: Vista de las detecciones (regiones) de las personas en las distintas cámaras. Abajo: La posición (puntos) de las personas proyectada en el plano del suelo.

Los datos necesarios para su ejecución, aparte del código y las bibliotecas, son los datasets (vídeos y matrices de calibración) y los ficheros del *groundtruth* (para la evaluación), así como los datos de las detecciones del detector pues actualmente éstas se leen de fichero, pero se puede modificar fácilmente para llamar a una función externa que analice una imagen dada. En cuanto a los datasets, se ha usado la función de OpenCV que permite cargar una fuente de imágenes de entrada desde cualquier origen, ya sea una cámara, un vídeo o una carpeta con los frames almacenados como es el caso. Si se requiere utilizar un dataset diferente se debe cambiar la función del módulo de *groundtruth* para que realice la lectura del sitio correspondiente, modificando la ubicación o el código disponible en el fichero *groundTruthParser.py*.

La carpeta *Debugging* contiene scripts que permiten observar los datos leidos del dataset, scripts de ejemplo de opencv u otras utilidades no necesarias para la ejecución del sistema. La carpeta *Utilities* contiene scripts con funciones y utilidades necesarias para la ejecución del sistema. En esta carpeta se encuentra, entre otros, el *wrapper* de OpenCV que permite ver el historial reciente de las imágenes mostradas.

Bibliografía

[1] Nicolas Schneider and Dariu M Gavrila. Pedestrian path prediction with recursive bayesian filters: A comparative study. In *German Conference on Pattern Recognition*, pages 174–183. Springer, 2013.

[2] Shoou-I Yu, Yi Yang, and Alexander Hauptmann. Harry potter's marauder's map: Localizing and tracking multiple persons-of-interest by nonnegative discretization. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3714–3720. IEEE, 2013.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition Conference (CVPR)*, volume 1, pages 886–893. IEEE, 2005.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[5] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[6] Ergys Ristani and Carlo Tomasi. Features for multi-target multi-camera tracking and re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6036–6046, 2018.

[7] Horesh Ben Shitrit, Jérôme Berclaz, François Fleuret, and Pascal Fua. Multi-commodity network flow for tracking multiple people. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1614–1627, 2014.

[8] Xinchao Wang, Engin Türetken, Francois Fleuret, and Pascal Fua. Tracking interacting objects using intertwined flows. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2312–2326, 2016.

[9] A. T. Kamal, J. H. Bappy, J. A. Farrell, and A. K. Roy-Chowdhury. Distributed multi-target tracking and data association in vision networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1397–1410, July 2016.

[10] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):267–282, 2008.

[11] Yuanlu Xu, Xiaobai Liu, Lei Qin, and Song-Chun Zhu. Cross-view people tracking by scene-centered spatio-temporal parsing. In *AAAI*, pages 4299–4305, 2017.

[12] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *Journal on Image and Video Processing*, 2008:1, 2008.

[13] Peter Jank, Karel Koplik, Tomáš Dulík, and Istvan Szabo. Comparison of tracking algorithms implemented in opencv. In *MATEC Web of Conferences 20th International Conference on Circuits, Systems, Communications and Computers (CSCC 2016)*. EDP Sciences, 2016.

[14] Jorge Andrés Galindo. Evaluación de plataformas de bajo coste para construir un sistema de vídeo-vigilancia. In *Trabajo Fin de Grado. Ing. Informática. Universidad de Zaragoza*, 2017.